AD

TECHNICAL REPORT ARCCB-TR-94033

ADAPTIVE FINITE ELEMENT METHOD I: SOLUTION ALGORITHM AND COMPUTATIONAL EXAMPLES

J.M. COYLE
J.E. FLAHERTY

AUGUST 1994

US ARMY ARMAMENT RESEARCH, DEVELOPMENT AND ENGINEERING CENTER
CLOSE COMBAT ARMAMENTS CENTER
BENÉT LABORATORIES
WATERVLIET, N.Y. 12189-4050

DISCLAIMER

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

The use of trade name(s) and/or manufacturer(s) does not constitute an official indorsement or approval.

DESTRUCTION NOTICE

For classified documents, follow the procedures in DoD 5200.22-M, Industrial Security Manual, Section II-19 or DoD 5200.1-R, Information Security Program Regulation, Chapter IX.

For unclassified, limited documents, destroy by any method that will prevent disclosure of contents or reconstruction of the document.

For unclassified, unlimited documents, destroy when the report is no longer needed. Do not return it to the originator.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1994	Final

4. TITLE AND SUBTITLE
ADAPTIVE FINITE ELEMENT METHOD I:
SOLUTION ALGORITHM AND COMPUTATIONAL EXAMPLES

5. FUNDING NUMBERS
PRON: 1A323F2TA11A
AMCMS: 612624H181100

6. AUTHOR(S)
J.M. Coyle and J.E. Flaherty

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army ARDEC
Benét Laboratories, SMCAR-CCB-TL
Watervliet, NY 12189-4050

8. PERFORMING ORGANIZATION REPORT NUMBER
ARCCB-TR-94033

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army ARDEC
Close Combat Armaments Center
Picatinny Arsenal, NJ 07806-5000

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

An adaptive finite element method is developed to solve initial boundary value problems for vector systems of parabolic partial differential equations in one space dimension and time. The differential equations are discretized in space using piecewise linear finite element approximations. Superconvergence properties and quadratic polynomials are used to derive a computationally inexpensive approximation to the spatial component of the error. This technique is coupled with time integration schemes of successively higher orders to obtain an approximation of the temporal and total discretization errors. These approximate errors are used to control an adaptive mesh refinement strategy. Refinement is performed in space, time, or both space and time depending on the dominant component of the error estimate. A computer code coupling this refinement strategy and stable mesh movement has been written and applied to a number of problems. These computations confirm that proper mesh movement can reduce the computational efforts associated with mesh refinement.

14. SUBJECT TERMS
Parabolic Differential Equations, Adaptive Finite Elements, Superconvergence, Error Estimation, Mesh Refinement, Mesh Movement

15. NUMBER OF PAGES
51

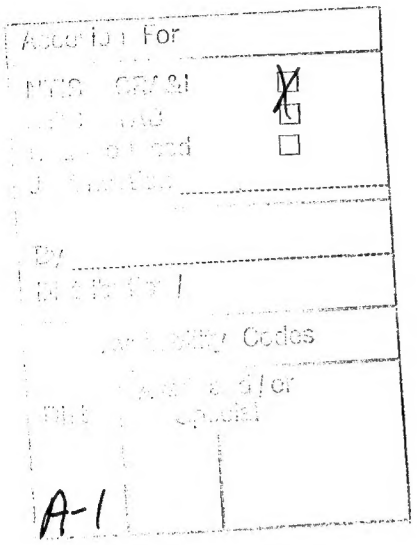
16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

Tables

List of Illustrations



INTRODUCTION

During the past decade and a half, a great deal of interest has arisen in the development of reliable, robust, and efficient software for the numerical solution of partial differential equations. The origin of this interest is obvious. Partial differential equations are used to mathematically model physical reality for many scientific disciplines. The complexity of these equations usually require that numerical methods be employed in order to obtain solutions. Most scientists and engineers prefer to use existing software for this process in order to avoid developing their own numerical techniques. In this way, they can concentrate on the interpretation of the computed results.

Before this time, even though the interest in solving partial differential equations numerically existed, computers were not up to the task of supporting general purpose packages along the lines of what existed for ordinary differential equations (cf., eg., Gear (ref 1) or Petzold (ref 2)). Due to restricted computing power, the tendency was to write problem specific computer codes. However, advances in computer hardware, computer systems, and numerical algorithms led to the feasibility of developing general purpose partial differential equation solvers (cf., e.g., Babuska, Chandra, and Flaherty (ref 3) and Flaherty, Paslow, Shephard, and Vasilakis (ref 4) for a sampling).

The major contribution that the field of numerical analysis made to this endeavor was the development of adaptive numerical methods (refs 3,4). An adaptive numerical method is a numerical method which automatically adjusts the discretization of the problem domain and/or the order of the method while the solution of the problem is being determined. This adjustment is performed so as to optimize the solution process in some way. This optimization usually takes the form of obtaining a solution with as coarse a discretization level and/or as low an order as possible for a given error tolerance. This criterion is not applied globally, however, but in a local manner. The result is that the same discretization and/or order is not used uniformly throughout the solution process, as is the case with ordinary numerical methods. Rather, finer discretizations and/or higher order methods are used in regions of the problem domain where numerical approximations are more difficult to make, and coarser discretizations and/or lower orders are used elsewhere. In this way, a reliable numerical solution is found as efficiently as possible, in terms of computer resources, and with minimal user interface.

Even with the advances in computer science, electrical engineering, and numerical analysis, it is still not practical to write a software package that attempts to solve all possible types of partial differential equations. At the very least, the three basic types of partial differential equations: elliptic, hyperbolic, and parabolic, demand three different numerical approaches in order to compute reliable solutions (cf., Lapidus and Pinder (ref 5)). The typical general purpose partial differential equation solver is still going to be limited in scope in some way.

This is the first in a series of four reports whose overall purpose is to describe an adaptive finite element method (AFEM) for solving systems of parabolic partial differential equations of the form

$$u_t + f(x,t,u,u_x) = [D(x,t,u)u_x]_x, a < x < b, t > 0, \quad (1)$$

subject to prescribed initial conditions and linear separated boundary conditions. The scalar variables x and t represent spatial and temporal coordinates and denote partial differentiation when used as subscripts; a and b are scalar constants; $u(x,t)$ and $f(x,t,u,u_x)$ are M-vectors; and $D(x,t,u)$ is an $M \times M$ matrix. The ultimate goal of AFEM is to determine an approximate solution to Eq. (1) to within a user prescribed error tolerance.

Equations of the form of Eq. (1) arise in many applications. When f is identically zero, Eq. (1) is a system of nonlinear heat or diffusion equations. They model physical circumstances as diverse as the temperature in a melting solid (cf., Friedman (ref 6)) and bacterial motion (cf., Keller and Odell (ref 7)). If f is a function of u only, then Eq. (1) is a system of reaction-diffusion equations. Such equations have been extensively studied as models of combustion (cf., Kapila (ref 8)), chemical reaction (cf., Fife (ref 9)), and population dynamics (cf., Hoppensteadt (ref 10)). In addition, if

$$f(u,u_x) = u^T u_x , \quad (2)$$

Eq. (1) arises in convective flows (cf., Batchelor (ref 11)). Therefore, even though this method does not address every variety of problem possible, it would still be a useful numerical tool in a large number of scientific areas.

Many technological situations, including some of those mentioned above, involve the rapid formation, propagation, and disintegration of small-scale structures. Some examples are shock waves in compressible flows, shear layers in laminar and turbulent flows, phase boundaries in nonequilibrium processes, combustion fronts, and classical boundary layers. It is this increasing complexity of the physical problem that scientists and engineers want to address, as well as the desire for reliable and robust software tools to accurately and efficiently describe these phenomena that led to the development of adaptive numerical techniques, in general, and AFEM, in particular.

The adaptive strategies utilized by AFEM are: (1) error estimation coupled with (2) local mesh refinement (cf., e.g., Adjerid and Flaherty (ref 12), Babuska and Dorr (ref 13), Babuska, Zienkiewicz, Gago, and Oliveira (ref 14), Bank and Weiser (ref 15), Berger and Oliger (ref 16), Bieterman and Babuska (refs 17,18), Moore and Flaherty (ref 19), Shephard, Qingxian, and Baehmann (ref 20), Strouboulis and Oden (ref 21), Zienkiewicz and Zhu (ref 22), and (3) mesh movement (cf., e.g., Adjerid and Flaherty (ref 12), Arney and Flaherty (ref 23), Bell and Shubin (ref 24), Davis and Flaherty (ref 25), Dorfi and Drury (ref 26), Dwyer (ref 27), Ewing, Russell, and Wheeler (ref 28), Hyman (ref 29), Kansa, Morgan, and Morris (ref 30), Miller and Miller (refs 31,32), Petzold (ref 33), Rai and Anderson (ref 34), Russell and Ren (ref 35), Saltzman and Brackbill (ref 36), Smooke and Koszykowski (ref 37), Thompson (ref 38), Verwer, Blom, Furzeland, and Zegeling (ref 39), and White (ref 40)). Detailed descriptions of how AFEM implements these adaptive strategies are found in separate technical reports by Coyle and Flaherty entitled: Adaptive Finite Element Method II: Error Estimation (ref 41); Adaptive Finite Element Method III: Mesh Refinement (ref 42); and Adaptive Finite Element Method IV: Mesh Movement (ref 43).

This report is an introduction to AFEM as well as an illustration of AFEM's usefulness as a computational tool. In particular, the purpose of this report is to formally present the equations that AFEM attempts to solve, describe the solution algorithm employed by AFEM, and present computational examples. Specifically, in the Problem Statement section, the problem that AFEM was developed to solve is formally stated and a weak form of the problem is derived. The Spatial Discretization and Temporal Discretization sections illustrate how AFEM numerically discretizes the weak form of the problem developed in the second section on a nonuniform, moving mesh. The Spatial Discretization section delineates the spatial discretization, demonstrating how AFEM utilizes both piecewise linear and quadratic finite elements. The Temporal Discretization section recounts the temporal discretization, depicting how both the backward Euler method and the trapezoidal rule are applied to the equations derived in the third section. Then a brief description of AFEM's adaptive procedures is given and the overall solution algorithm is presented. The Example section presents computational results for four test problems chosen in order to illustrate different aspects of the code. Finally, the last section contains a summary and discussion of this report.

PROBLEM STATEMENT

Consider the problem of finding a numerical solution of an M-dimensional system of partial differential equations of the form

$$\boldsymbol{u}_t(x,t) + \boldsymbol{f}(x,t,u,u_x) = [\boldsymbol{D}(x,t,u)\boldsymbol{u}_x(x,t)]_x \ , \ a<x<b \ , \ t>0 \, , \tag{3a}$$

subject to the initial conditions

$$\boldsymbol{u}(x,0) = \boldsymbol{u}^0(x) \ , \ a \le x \le b \tag{3b}$$

and linear separated boundary conditions

$$\begin{aligned} \boldsymbol{A}^l(t)\boldsymbol{u}(a,t) + \boldsymbol{B}^l(t)\boldsymbol{u}_x(a,t) &= \boldsymbol{g}^l(t) \\ \boldsymbol{A}^r(t)\boldsymbol{u}(b,t) + \boldsymbol{B}^r(t)\boldsymbol{u}_x(b,t) &= \boldsymbol{g}^r(t) \end{aligned} \ , \ t>0 \ . \tag{3c}$$

The variables x and t represent spatial and temporal coordinates and denote partial differentiation when they are used as subscripts; $\boldsymbol{u}$, $\boldsymbol{f}$, $\boldsymbol{u}^0$, $\boldsymbol{g}^l$, and $\boldsymbol{g}^r$ are M-vectors; and $\boldsymbol{D}$, $\boldsymbol{A}^l$, $\boldsymbol{B}^l$, $\boldsymbol{A}^r$, and $\boldsymbol{B}^r$ are $M \times M$ matrices.

The problem is assumed to be well-posed and parabolic; thus, e.g., $\boldsymbol{D}(x,t,u)$ is positive definite. The rows of $\boldsymbol{B}^l$ and $\boldsymbol{B}^r$ are restricted to be either entirely zero or a row of the $M \times M$ identity matrix. When the i^{th} row of $\boldsymbol{B}^l$ or $\boldsymbol{B}^r$ is identically zero, then A^l_{ii} or A^r_{ii} cannot be zero, respectively, and the boundary condition is a Dirichlet (essential) condition. Otherwise, the boundary condition is a Neumann or Robbins (natural) condition.

A weak form of the problem is constructed by multiplying Eq. (3a) by a test function $v(x,t) \in H^1_0$, integrating the result with respect to x from a to b, and integrating the diffusive term by parts to obtain

$$(v,u_t) + (v,f) + A(v,u) = v^T D u_x|_a^b \ , \ t>0 \ , \ \textit{for all } v \in H_0^1 \ . \tag{4a}$$

The inner product (v,u) and strain energy $A(v,u)$ are defined as

$$(v,u) = \int_a^b v^T u \ dx \ , \ A(v,u) = \int_a^b v_x^T D u_x \ dx \ . \tag{4b,c}$$

Functions v belonging to H^1 are required to have finite values of (v,v) and (v_x,v_x). Functions in H_0^1 are in H^1 and vanish at $x = a$ and/or b if an essential boundary condition is applied there. Any weak solution $u \in H_E^1$ of Eq. (4a) must also satisfy any essential boundary conditions at $x = a$

$$u_i(a,t) = \left[g_i^l(t) - \sum_{\substack{j=1 \\ j\neq i}}^{M} A_{ij}^l(t) u_j(a,t) \right] \div A_{ii}^l(t) \tag{4d}$$

or at $x = b$

$$u_i(b,t) = \left[g_i^r(t) - \sum_{\substack{j=1 \\ j\neq i}}^{M} A_{ij}^r(t) u_j(b,t) \right] \div A_{ii}^r(t) \tag{4e}$$

when the i^{th} row of B^l and/or B^r is zero, respectively. Natural boundary conditions replace the i^{th} component of u_x at $x = a$ or b in Eq. (4a) when prescribed.

Initial conditions for Eq. (4a) are obtained by L^2 projection, i.e.,

$$(v,u) = (v,u^0) \ , \ t = 0 \ , \ \textit{for all } v \in H_0^1 \ . \tag{4f}$$

A discrete version of the weak system Eq. (4) is constructed by using finite element-Galerkin procedures in space and finite difference techniques in time on a fully adaptive mesh (one that is both refined and moved as time progresses).

SPATIAL DISCRETIZATION

To discretize Eq. (4a) in space, introduce a time-dependent partition

$$\Pi_N(t) = \{ \ a = x_0 < x_1(t) < x_2(t) < \ldots < x_N = b \ \} \tag{5}$$

of (a,b) into N subintervals $(x_{i-1}(t), x_i(t))$, $i=1,2,\ldots,N$ and approximate u and v by piecewise polynomial functions U and V, respectively, with respect to this partition. Thus, the spatially-discrete form of Eq. (4a) consists of finding $U \in S_E^N \subset H_E^1$ such that

$$(V,U_t) + (V,f) + A(V,U) = V^T D U_x \Big|_a^b \ , \ t>0 \ , \qquad \textbf{(6a)}$$
$$\textit{for all } V \in S_0^N \subset H_0^1 \ ,$$

$$(V,U) = (V,u^0) \ , \ t=0 \ , \textit{ for all } V \in S_0^N \subset H_0^1 \ . \qquad \textbf{(6b)}$$

The spaces S_E^N and S_0^N will consist of either piecewise linear or piecewise quadratic polynomial functions. The spaces of piecewise linear polynomials are denoted $S_E^{N,1}$ and $S_0^{N,1}$ and a basis is easily constructed in terms of the familiar "hat" functions

$$\phi_i(x,t) = \begin{cases} \dfrac{x - x_{i-1}(t)}{x_i(t) - x_{i-1}(t)} \ , \ x_{i-1}(t) \le x \le x_i(t) \\ \dfrac{x_{i+1}(t) - x}{x_{i+1}(t) - x_i(t)} \ , \ x_i(t) \le x \le x_{i+1}(t) \ . \\ 0 \qquad , \textit{ otherwise} \end{cases} \qquad \textbf{(7)}$$

The piecewise linear finite element solution $U_1 \in S_E^{N,1}$ is written in the form

$$U_1(x,t) = \sum_{i=0}^{N} c_i(t)\phi_i(x,t) \qquad \textbf{(8)}$$

and determined by solving the ordinary differential system

$$(V_1,U_{1_t}) + (V_1,f(\cdot,t,U_1,U_{1_x})) + A(V_1,U_1) = V_1^T D U_{1_x} \Big|_a^b \ , \qquad \textbf{(9a)}$$
$$t>0 \ , \textit{ for all } V_1 \in S_0^{N,1} \ ,$$

$$(V_1,U_1) = (V_1,u^0) \ , \ t=0 \ , \textit{ for all } V_1 \in S_0^{N,1} \ , \qquad \textbf{(9b)}$$

where the piecewise linear test functions $V_1 \in S_0^{N,1}$ have a form similar to Eq. (8).

Piecewise quadratic approximations $U_2 \in S_0^{N,2}$ are constructed by adding a "hierarchical" correction $E_2(x,t)$ to U_1, i.e.,

$$U_2(x,t) = U_1(x,t) + E_2(x,t) \ , \qquad \textbf{(10a)}$$

where

$$E_2(x,t) = \sum_{i=1}^{N} d_{i-\frac{1}{2}}(t)\,\psi_{i-\frac{1}{2}}(x,t) \ . \tag{10b}$$

The basis $\psi_{i-\frac{1}{2}}(x,t)$, $i=1,2,...,N$, for the quadratic correction has the form

$$\psi_{i-\frac{1}{2}}(x,t) = \begin{cases} \dfrac{2[x-x_{i-1}(t)][x-x_i(t)]}{[x_i(t)-x_{i-1}(t)]^2} & , \ x_{i-1}(t) < x < x_i(t) \\ 0 & , \ \textit{otherwise} \end{cases} \ . \tag{11}$$

Piecewise quadratic solutions are determined by solving

$$(V_2,U_{2_t}) + (V_2,f(\cdot,t,U_2,U_{2_x})) + A(V_2,U_2) = V_2^T D U_{2_x}\big|_a^b \ , \quad t>0 \ , \ \textit{for all} \ V_2 \in S_0^{N,2} \ , \tag{12a}$$

$$(V_2,U_2) = (V_2,u^0) \ , \ t=0 \ , \ \textit{for all} \ V_2 \in S_0^{N,2} \ , \tag{12b}$$

where V_2 has a form similar to Eq. (10).

TEMPORAL DISCRETIZATION

Discretization in time is performed by integrating, for example, Eq. (6a) over the time step from t^{n-1} to t^n to obtain

$$\int_{t^{n-1}}^{t^n} (V,U_t) + (V,f) + A(V,U)\ dt = \int_{t^{n-1}}^{t^n} V^T D U_x\big|_a^b \ dt \ , \quad \textit{for all} \ V \in S_0^N \ , \tag{13a}$$

$$(V,U) = (V,u^0) \ , \ t=0 \ , \ \textit{for all} \ V \in S_0^N \ . \tag{13b}$$

Rewriting the left-hand side of Eq. (13) to emphasize the contribution over each subinterval gives

$$\sum_{i=1}^{N} \int_{t^{n-1}}^{t^n} \int_{x_{i-1}}^{x_i} V^T U_t + V^T f + V_x^T D U_x \, dx \, dt = \int_{t^{n-1}}^{t^n} V^T D U_x \Big|_a^b \, dt \,, \quad \text{for all } V \in S_0^N \,, \tag{14a}$$

$$(V,U) = (V,u^0) \,, \; t=0 \,, \text{ for all } V \in S_0^N \,. \tag{14b}$$

The integration in Eq. (14) will be over an irregular region due to the mesh motion with the test function V having an undesirable time dependency.

In order to overcome this difficulty, introduce a linear transformation

$$x = x_{i-1}^{n-1} + \left(x_{i-1}^n - x_{i-1}^{n-1}\right)\tau + \tfrac{1}{2}\Delta x_i^{n-1}(1+\xi) + \tfrac{1}{2}(\Delta x_i^n - \Delta x_i^{n-1})\tau(1+\xi), \tag{15a}$$

$$t = t^{n-1} + \Delta t^n \tau \tag{15b}$$

where

$$\Delta x_i^n = x_i^n - x_{i-1}^n \,, \; \Delta t^n = t^n - t^{n-1} \tag{15c,d}$$

and

$$x_i^n = x_i(t^n) \,, \; i = 0,1,\ldots,N \,. \tag{15e}$$

This transformation maps the rectangle $\{(\xi,\tau) | -1 \le \xi \le 1, 0 \le \tau \le 1\}$ onto the trapezoidal space-time element whose vertices are

$$(x_{i-1}^{n-1}, t^{n-1}) \,, \; (x_i^{n-1}, t^{n-1}) \,, \; (x_{i-1}^n, t^n) \text{ and } (x_i^n, t^n) \,.$$

The basis elements $\phi_{i-1}(x,t)$ and $\phi_i(x,t)$, the only nonzero ones on $\{(x,t) | x_{i-1}(t) \le x \le x_i(t) \,, \; t^{n-1} \le t \le t^n\}$, are transformed to functions with no τ dependency; thus, $\phi_{i-1}(x,t)$ and $\phi_i(x,t)$ become, respectively,

$$\hat{\phi}_{-1}(\xi) = \tfrac{1}{2}(1-\xi) \,, \text{ and} \tag{16a}$$

$$\hat{\phi}_1(\xi) = \tfrac{1}{2}(1+\xi) \,, \; -1 \le \xi \le 1 \,. \tag{16b}$$

Define

$$F_i = \int_{t^{n-1}}^{t^n} \int_{x_{i-1}}^{x_i} (V^T U_t + V^T f + V_x^T D U_x)\, dx\, dt \tag{17}$$

and write Eq. (14) as

$$\sum_{i=1}^{N} F_i = \int_{t^{n-1}}^{t^n} V^T D U_x|_a^b \, dt \ , \ for\ all\ V \in S_0^N \ , \tag{18a}$$

$$(V,U) = (V,u^0) \ , \ t=0 \ , \ for\ all\ V \in S_0^N \ . \tag{18b}$$

by substituting Eq. (17) into Eq. (14). Equation (18a) can also be rewritten as

$$\sum_{i=1}^{N} F_i = \int_0^1 V^T D U_x|_a^b \, \Delta t^n d\tau \ , \ for\ all\ V \in S_0^N \ , \tag{18c}$$

$$(V,U) = (V,u^0) \ , \ t=0 \ , \ for\ all\ V \in S_0^N \ , \tag{18d}$$

by performing the change of variables from t to τ (cf., Eq. (15b)).

Transforming Eq. (17) from the $(x\text{-}t)$-plane to the (ξ,τ)-plane (cf., Eq. (15)) yields

$$F_i = \int_0^1 \int_{-1}^1 \left[V^T \left(U_\tau \frac{1}{t_\tau} - U_\xi \frac{x_\tau}{x_\xi t_\tau} \right) + V^T f + V_\xi^T D U_\xi \frac{1}{(x_\xi)^2} \right] x_\xi t_\tau \, d\xi \, d\tau \tag{19a}$$

which reduces to

$$F_i = \int_0^1 \int_{-1}^1 \left[V^T (U x_\xi)_\tau - V^T (U \dot{x})_\xi t_\tau + V^T f x_\xi t_\tau + V_\xi^T D U_\xi \frac{t_\tau}{x_\xi} \right] d\xi \, d\tau \tag{19b}$$

where

$$\dot{x} = \frac{x_\tau}{t_\tau} \ . \tag{19c}$$

Equation (19b) can be simplified further by noting that

$$t_\tau = \Delta t^n \tag{20a}$$

and integrating by parts to obtain

$$F_i = G_i(1) - G_i(0) + \Delta t^n \int_0^1 I_i(\tau)\, d\tau \tag{20b}$$

where

$$G_i(\tau) = \int_{-1}^{1} V^T U x_\xi \, d\xi \ , \tag{20c}$$

$$I_i(\tau) = \int_{-1}^{1} \left[-V^T (U\dot{x})_\xi + V^T f x_\xi + V_\xi^T D U_\xi \frac{1}{x_\xi} \right] d\xi \ . \tag{20d}$$

Substituting Eq. (20b) into Eq. (18c) then yields

$$\sum_{i=1}^{N} \left[G_i(1) - G_i(0) + \Delta t^n \int_0^1 I_i(\tau) d\tau \right] = \Delta t^n \int_0^1 V^T D U_x \Big|_a^b \, d\tau \ , \quad \textit{for all } V \in S_0^N \ , \tag{21a}$$

$$(V,U) = (V,u^0) \ , \ t=0 \ , \textit{ for all } V \in S_0^N \ . \tag{21b}$$

All that remains is to approximate the time integrals in Eq. (21) using a quadrature rule. This is done by using a weighted two-step method, which for Eq. (21) has the form

$$\sum_{i=1}^{N} \left[G_i(1) + G_i(0) + \Delta t^n \theta I_i(1) + \Delta t^n (1-\theta) I_i(0) \right] = \Delta t^n \theta V^T D U_x \Big|_a^b \Big|_{\tau=1} + \Delta t^n (1-\theta) V^T D U_x \Big|_a^b \Big|_{\tau=0} , \textit{for all } V \in S_0^N , \theta \in [0,1] , \tag{22a}$$

$$(V,U) = (V,u^0) \ , \ t=0 \ , \textit{ for all } V \in S_0^N \ . \tag{22b}$$

Only two choices of θ are utilized: either $\theta = 1$, which yields the backward Euler method, or $\theta = \frac{1}{2}$, which yields the trapezoidal rule.

ADAPTIVE AND SOLUTION ALGORITHMS

The principle adaptive feature of AFEM is the incorporation of a mesh moving or an r-refinement scheme (r for redistribution). The basic idea of mesh movement is to initially place a fixed number of mesh points in optimal locations and then move them as the problem is solved so that their locations remain optimal.

AFEM's mesh moving algorithm is based on equidistribution, made popular by De Boor (ref 44) for problems in functional approximation. The goal of equidistribution, at least in the static case of functional analysis, is to determine a mesh so that a particular quantity is uniform over each subinterval. In this case, it has been shown (cf., De Boor (ref 44)) that the task of selecting a mesh to minimize the maximum discretization error is asymptotically equivalent (for a dense mesh) to equidistributing the local discretization error.

In order to facilitate the use of a static equidistribution law in a method for solving time-dependent problems, a few adjustments are made. First, the equidistribution law is extended to include a time dependency. Next, the resultant equation is approximately normalized. Finally, the normalized equation is differentiated with respect to time (cf., Coyle and Flaherty (ref 43)).

The result of the above process is a stable ordinary differential equation which represents a dynamic equidistribution law. This dynamic law no longer demands that some quantity be uniform over each subinterval, however. Instead, a time-dependent mesh is determined such that the same quantity as in the static case now remains constant for all time over each subinterval.

Mesh movement was the first adaptive technique developed for time-dependent partial differential equations because it does not demand as much computer resources as some other adaptive techniques. No extra calculations to determine error approximations need be made nor extra storage need be created to accommodate additional mesh points and the data structures are simple.

Mesh movement does have its drawbacks, however. If sufficient care is not exercised, mesh trajectories can leave the domain of definition, cross each other, or oscillate wildly from time step to time step (cf., Coyle, Flaherty, and Ludwig (ref 45)).

Another drawback to mesh movement is that it does not guarantee that the approximate solution satisfies a prescribed accuracy tolerance for any discretization level. As a result, most researchers have either abandoned moving in favor of other adaptive techniques that guarantee such accuracy or incorporated such methods into their moving codes, as was done with AFEM.

The additional adaptive technique incorporated into AFEM is a local or h-refinement method coupled with a posteriori error estimation. A local refinement method is one that determines subregions of the domain where the solution has not been adequately approximated and solves the problem again using a finer discretization level in these subregions. This determination is usually made by estimating the error of the numerical approximation throughout the entire domain. Local refinement methods for time-dependent partial differential equations do not only refine the discretization, but also coarsen it in regions where a fine discretization is unnecessary. In this way, the methods attempt to lessen the burden associated with solving the

problem more than once in some region.

AFEM's refinement strategy estimates the total discretization error as well as its spatial and temporal components. Then the type of refinement that occurs is dependent upon which error component is dominant. If the temporal error dominates, then a new global time step is calculated and the problem is solved again using this time step. If the spatial component of the error is the dominant one, then more mesh points are added (and/or deleted) in regions where the estimate is high (and/or low) and the problem is solved again with the same time step or a larger one if the temporal error estimate is sufficiently small. If no component dominates, and yet the total error estimate is too large, then refinement occurs in both space and time (cf., Coyle and Flaherty (ref 42)).

A posteriori error estimates are obtained by effectively solving the problem more than once, using higher order methods, and then finding the difference between the higher order and the lower order solutions. Hierarchial formulation and nodal superconvergence (cf., Adjerid and Flaherty (ref 12)) are used to reduce the computational complexity of the error estimation procedure (cf., Coyle and Flaherty (ref 41)).

The solution algorithm for solving Eq. (1) consists of dividing the temporal region into strips $0 = t^0 < t^1 < \ldots$. Suppose that a solution has been successfully determined up to some time level, t^{n-1}. The goal, then, is to compute a solution with a given user tolerance, TOL, in the H_1 norm at the next time level t^n.

The first step is to advance the mesh using the dynamic equidistribution law to a time level

$$t^n = t^{n-1} + \Delta t^{n-1} \tag{23}$$

where Δt^{n-1} is the last time step used to successfully determine the solution at time level, t^{n-1}. Next, Eq. (1) is discretized in space using Galerkin's method with piecewise linear finite element approximations (cf., Eq. (9)). Temporal discretization is performed by the Backward Euler method (cf., Eq. (22) with $\theta = 1$). A second solution is calculated using trapezoidal rule integration (cf., Eq. (22) with $\theta = \frac{1}{2}$) and the difference between the two solutions is used to furnish the temporal error estimate. A third solution is obtained using quadratic finite elements (cf., Eq. (12)) and the trapezoidal rule (cf., Eq. (22) with $\theta = \frac{1}{2}$). The difference between this third solution and the original piecewise linear Backward Euler solution is used to furnish an estimate to the total error. The difference between this third solution and the piecewise linear trapezoidal rule solution provides the spatial error estimate. The total error estimate is now compared to TOL. If the error estimate exceeds TOL, spatial and/or temporal h-refinement is performed until the total error estimate at the resultant time level is less than TOL.

EXAMPLES

In this section we examine the performance of the adaptive finite element method described in the previous sections on four examples. In each example, we try to concentrate on a particular facet of the method as well as illustrate the method's dependence on some different input parameters available to the user.

Example 1

Consider a problem for Burgers' equation:

$$u_t + uu_x = \epsilon u_{xx} \ , \ 0 < x < 1 \ , \ t > 0 \qquad \textbf{(24a)}$$

$$u(x,0) = \sin \pi x \ , \ 0 \le x \le 1 \ , \qquad \textbf{(24b)}$$

$$u(0,t) = u(1,t) = 0 \ , \ t > 0 \ . \qquad \textbf{(24c,d)}$$

where $\epsilon = 5\times10^{-3}$. The solution of this problem is a pulse that steepens as it travels to the right until it forms a shock layer at $x = 1$. After a time of $O(\epsilon^{-1})$, the pulse dissipates and the solution decays to zero (cf., Figure 1a at the end of Example 1).

This problem was solved for a value of TOL equal to 0.8 but with different initial values of N and Δt. In Table 1, a summary of the results is presented when N and Δt are initially 100 and 0.1, respectively. Table 2 presents a summary for an initial N of 20 and an initial Δt of 0.02. Table 3 summarizes results when N and Δt are initially 50 and 0.1. Mesh trajectories corresponding to the data presented in Tables 1, 2, and 3 are shown in Figures 1b, 2, and 3, respectively (at the end of Example 1). The horizontal lines in these figures are used to distinguish the regions where the values of Δt are different.

These tables present the relevant refinement data for the significant time levels of the solution process. By a "significant time level," we mean a time level at which a refinement has occurred and one immediately preceding a refinement. The tables are divided into four columns labelled "Significant Times," "N" (with the initial N in parentheses), "Δt" (with the initial Δt in parentheses), and "Error Estimate," respectively. The significant time levels are found in the first column. Reading across the tables, one finds the values of N and Δt necessary to complete the solution step to within the given tolerance. A dash is used to indicate no change in N or Δt from the preceding row. The last column lists the value of the total error estimate at the successful completion of the time step. The last row of each table contains results for the final time of 0.8. In this manner, the tables outline the refinement process giving some indication as to why refinement occurred as well as demonstrating the success of the process.

Table 1 **Numerical Parameters for Solving Eqs. (24) with TOL = 0.8 and N and Δt Initially 100 and 0.1, Respectively**

Significant Times	N (100)	Δt (0.1)	Error Estimate
0.1000	-	-	0.2272
0.1619	-	0.0619	0.2161
0.2238	-	-	0.5470
0.2713	164	0.0475	0.3692
0.3664	-	-	0.4848
0.3970	-	0.0362	0.3693
0.8000	-	-	0.1606

Table 2 **Numerical Parameters for Solving Eqs. (24) with TOL = 0.8 and N and Δt Initially 20 and 0.02, Respectively**

Significant Times	N (20)	Δt (0.02)	Error Estimate
0.24	-	-	0.5094
0.26	40	-	0.2832
0.30	-	-	0.5806
0.32	69	-	0.1973
0.80	-	-	0.2901

Table 3 **Numerical Parameters for Solving Eqs. (24) with TOL = 0.8 and N and Δt Initially 50 and 0.1, Respectively**

Significant Times	N (50)	Δt (0.1)	Error Estimate
0.2000	-	-	0.3127
0.3000	75	-	0.2292
0.4000	-	-	0.3613
0.4572	-	0.0572	0.2522
0.8000	-	-	0.1389

In Adaptive Finite Element Method III (ref 42), Eqs. (24) are solved for a value of TOL = 0.8 and the same initial values of N and Δt as above. Only h-refinement was employed in Reference 42, however, since the purpose of that report is to introduce and discuss our refinement procedure. In Tables 4, 5, and 6, the results for solving Eqs. (24) using only our refinement scheme is presented. This is done in order to study the effects of our mesh moving scheme when coupled with refinement.

Table 4 **Numerical Parameters for Solving Eqs. (24) with TOL = 0.8 and N and Δt Initially 100 and 0.1, Respectively**

Significant Times	N (100)	Δt (0.1)	Error Estimate
0.1000	-	-	0.3221
0.1566	-	0.0566	0.2218
0.2131	-	-	0.5436
0.2549	165	0.0417	0.3455
0.8000	-	-	0.2728

Table 5 **Numerical Parameters for Solving Eqs. (24) with TOL = 0.8 and N and Δt Initially 20 and 0.02, Respectively**

Significant Times	N (20)	Δt (0.02)	Error Estimate
0.24	-	-	0.6741
0.26	46	-	0.3710
0.28	-	-	0.4330
0.30	66	-	0.3072
0.38	-	-	0.6210
0.40	109	-	0.2488
0.80	-	-	0.1595

Table 6 **Numerical Parameters for Solving Eqs. (24) with TOL = 0.8 and N and Δt Initially 50 and 0.1, Respectively**

Significant Times	N (50)	Δt (0.1)	Error Estimate
0.1000	-	-	0.3263
0.1573	62	0.0573	0.2227
0.2147	-	-	0.4766
0.2566	81	0.0419	0.3913
0.3405	-	-	0.4662
0.3824	118	-	0.3448
0.8000	-	-	0.1644

Comparison of the above tables is very encouraging. Consider for example, the case where N and Δt are initially 20 and 0.02, respectively (cf., Tables 2 and 5). In both instances, the initial time step was small enough so that no temporal refinement was necessary, and the code recognized this fact in each instance. However, when moving was coupled with refinement, only two refinements occurred and only 69 elements were ultimately needed to solve the problem

within a tolerance of 0.8 (cf., Table 2). For h-refinement, three refinements were necessary and 109 elements ultimately were used (cf., Table 5). This is a significant reduction in the necessary level of discretization, as we hoped would be accomplished by the incorporation of mesh movement.

Comparing the case where N and Δt are initially 50 and 0.1, respectively, is similar (cf., Tables 3 and 6). When moving was coupled with refinement, only two refinements occurred, only 75 elements were ultimately needed, and only a time step of 0.0572 was finally necessary to solve the problem within a tolerance of 0.8 (cf., Table 3). For h-refinement, three refinements were necessary (two of them in both space and time), 118 elements ultimately had to be used, and the final time step employed was 0.0419 (cf., Table 6). Once again, a significant reduction in the necessary level of discretization was accomplished by the incorporation of mesh movement.

It is only in the case where N and Δt are initially 100 and 0.1, respectively, that mesh movement does not improve the computation (cf., Table 1 and Table 4). When moving was coupled with refinement, three refinements were necessary (one of them in both space and time), 164 elements ultimately had to be used, and the final time step employed was 0.0362 (cf., Table 3). In the h-refinement case, two refinements were necessary (one of them in both space and time), 165 elements ultimately had to be used, and the final time step employed was 0.0417 (cf., Table 6). No significant reduction in the necessary level of discretization was accomplished by the incorporation of mesh movement. As a matter of fact, it could be argued that mesh movement performed worse since an extra temporal refinement step was necessary, while employing only one less element. This is not discouraging since, clearly, too many elements were used in the initial mesh; hence, it is difficult for a moving or redistribution scheme to make any dramatic improvements.

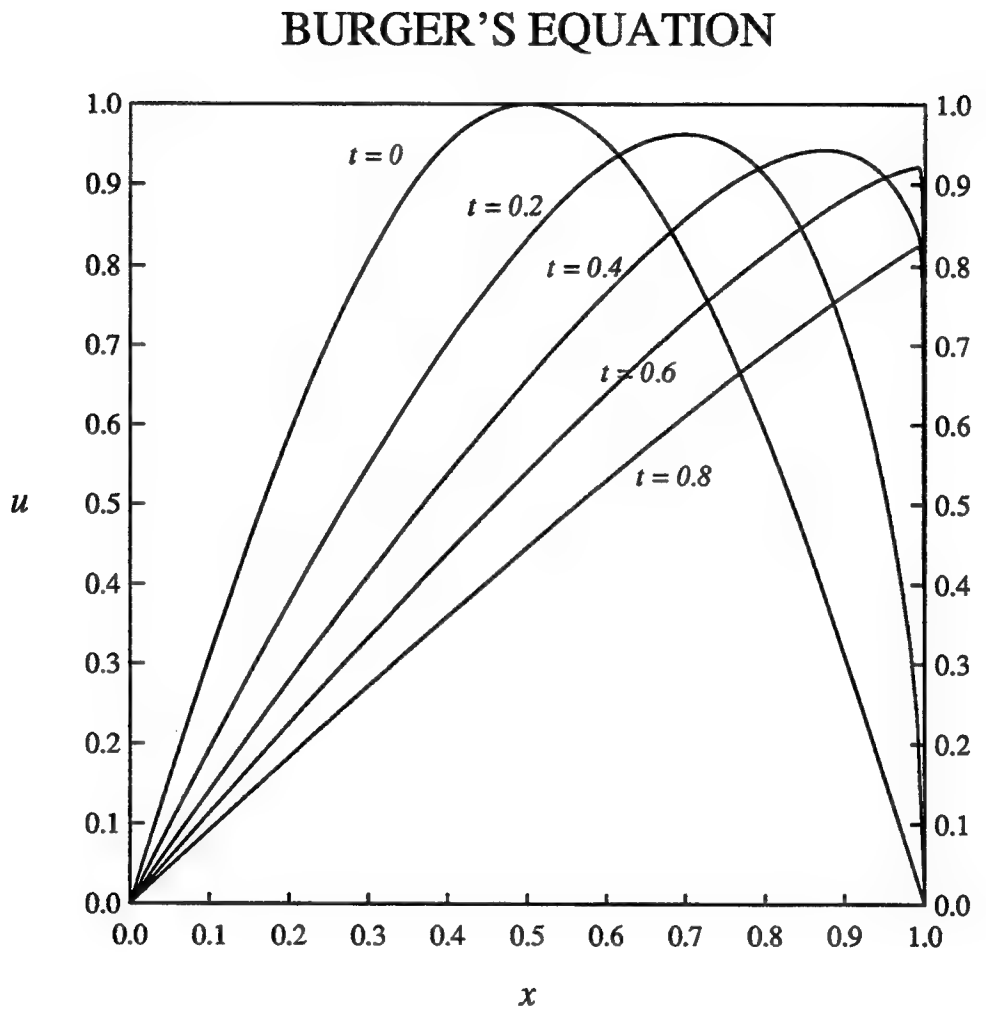


Figure 1a. Solutions to Example 1 for several instances of time.

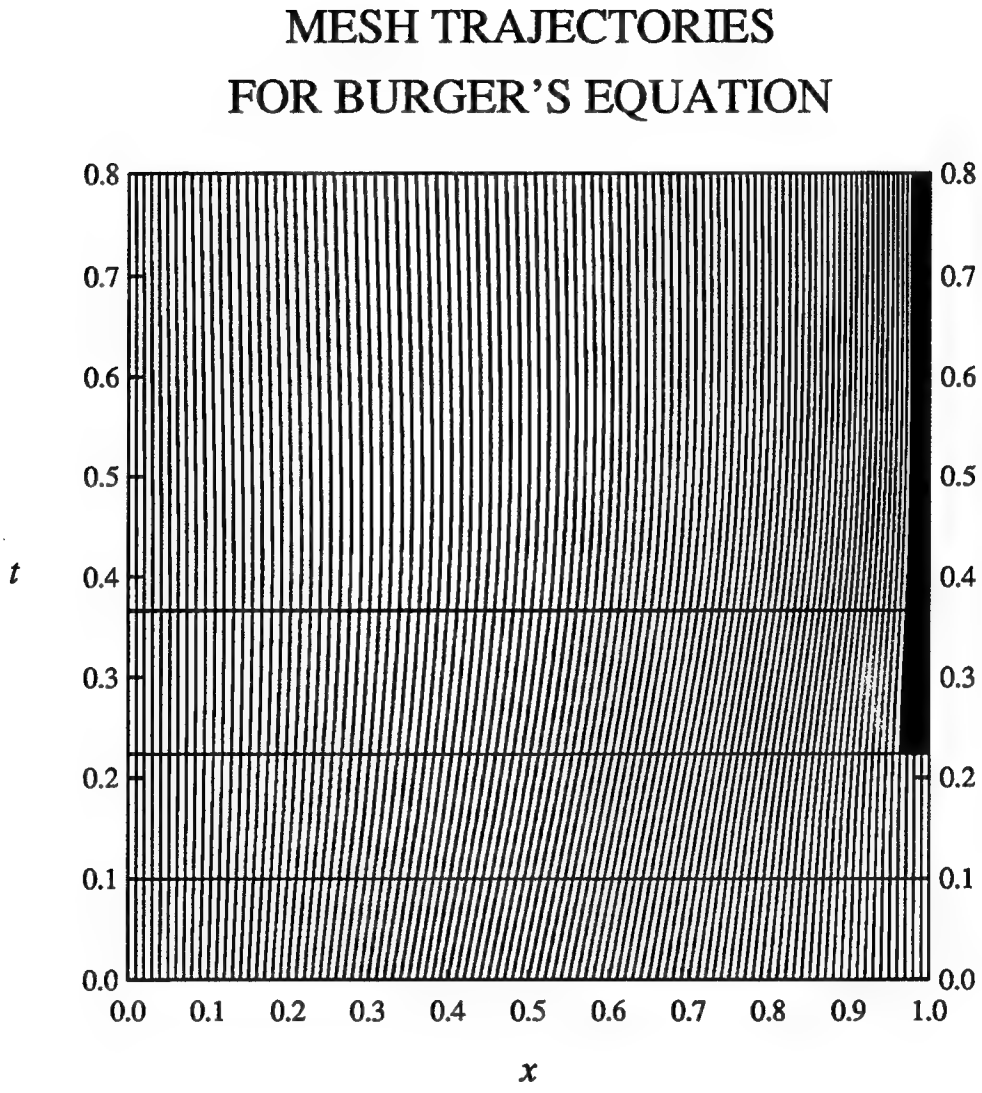


Figure 1b. Mesh trajectories for Example 1 corresponding to Table 1.

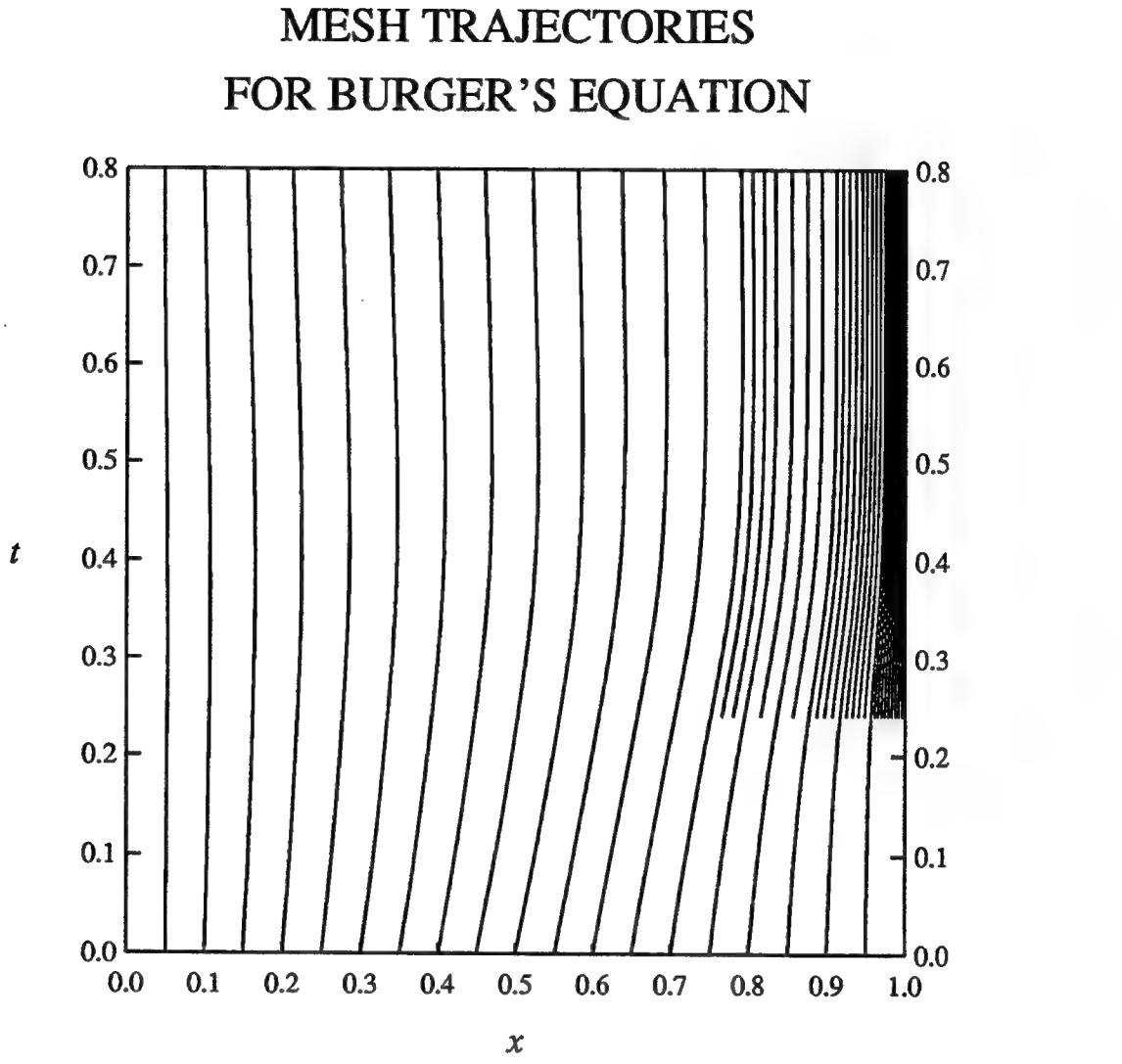


Figure 2. Mesh trajectories for Example 1 corresponding to Table 2.

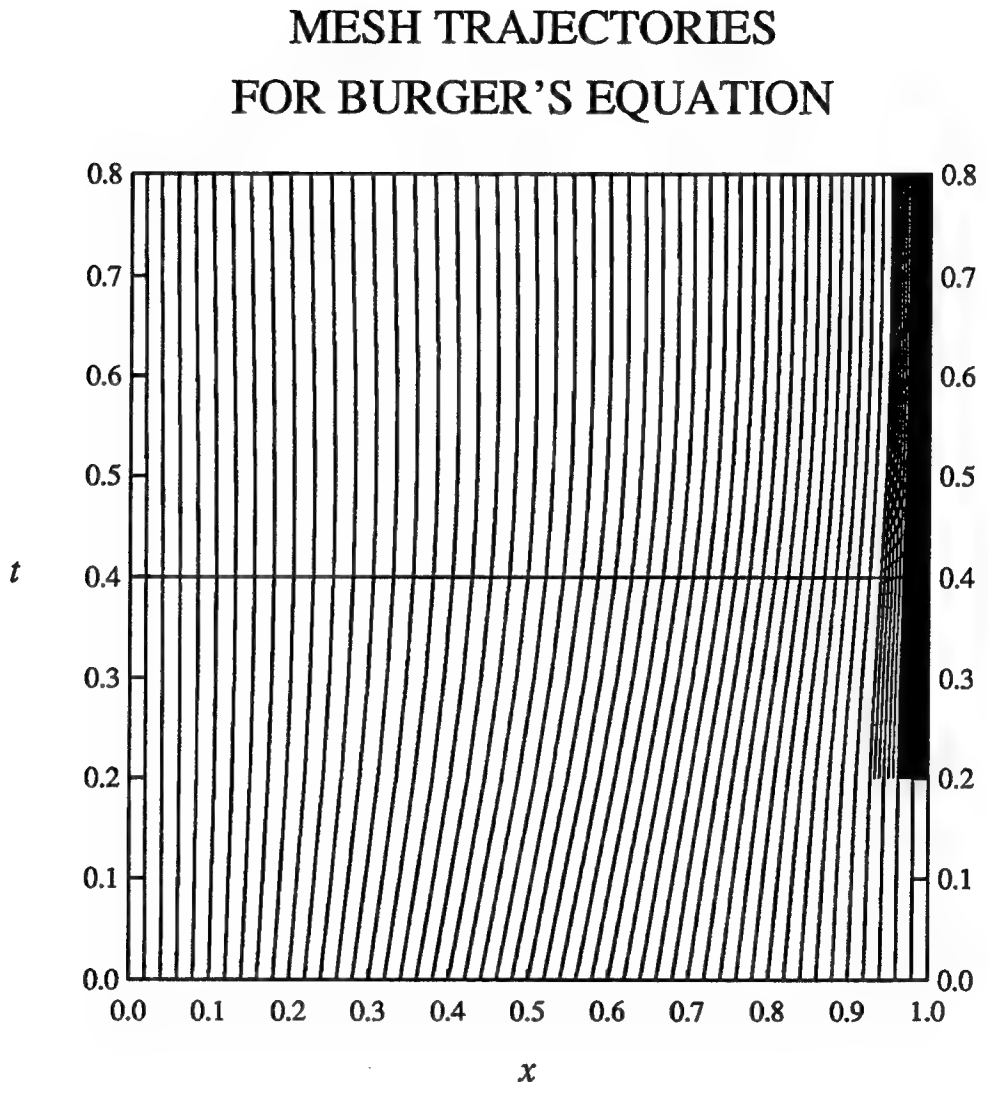


Figure 3. Mesh trajectories for Example 1 corresponding to Table 3.

Example 2

Consider the following partial differential equation

$$u_t + f(x,t,u,u_x) = \frac{1}{2r_1^2} u_{xx}, \quad 0 < x < 1, \quad t > 0. \tag{25a}$$

The initial and boundary conditions and the function f are chosen such that

$$u(x,t) = \tanh r_1[(x-1) + r_2 t] \tag{25b}$$

is the exact solution. Function (25b) is a wave that travels in the negative x direction when r_1 and r_2 are positive (cf., Figure 4 at the end of Example 2). The values of r_1 and r_2 determine the steepness of the wave and its propagation speed. We solved this problem for values of r_1 and r_2 equal to 10 and 1, respectively, in order to study the effect of a moving phenomenon on our adaptive algorithms. In particular, two different values of the error tolerance parameter, TOL, were selected to see if the adaptivity adjusted accordingly. An initial value of N equal to 10 and Δt equal to 0.1 were used in both instances. Table 7 summarizes the results for a value of TOL equal to 0.8, while Table 8 presents results when TOL was set to 0.5. Both tables are similar to the tables presented in Example 1. Mesh trajectories corresponding to the data presented in Tables 7 and 8 are shown in Figures 5 and 6, respectively (at the end of Example 2). The horizontal lines in these figures are used to distinguish the regions where the values of Δt are different.

Table 7 **Numerical Parameters for solving Eqs. (25) with TOL = 0.8 and N and Δt Initially 10 and 0.1, Respectively**

Significant Times	N (10)	Δt (0.1)	Error Estimate
0.0545	18	0.0545	0.5007
0.0893	27	0.0347	0.5197
0.2628	-	-	0.5933
0.2975	48	-	0.4456
1.0000	-	-	0.3205

Table 8 **Numerical Parameters for Solving Eqs. (25) with TOL = 0.5 and N and Δt Initially 10 and 0.1, Respectively**

Significant Times	N (10)	Δt (0.1)	Error Estimate
0.0394	23	0.0394	0.3003
0.0638	-	0.0243	0.3355
0.0830	-	0.0193	0.3140
0.2371	-	-	0.3858
0.2564	43	-	0.2783
0.4297	-	-	0.4189
0.4490	69	-	0.2803
0.6994	-	-	0.4297
0.7187	103	-	0.1850
1.0000	-	-	0.3118

As expected, the smaller the tolerance the more work the code must perform in order to determine an approximate solution to within that tolerance (cf., Tables 7 and 8). In particular, notice how often the time step is adjusted at the beginning of the solution process for both tolerance levels. The initial time step of 0.1 is a rather coarse value and the code does recognize this fact and reacts accordingly. However, a better strategy would seem to be a smaller initial Δt.

Until this point, all examples have been performed on an adaptive mesh that initially was uniform. This is not the only option available to a user of this code. The software will also accept a user determined mesh or the software will determine an equidistributed mesh (cf., Adaptive and Solution Algorithms section) using the initial data. The attempt that the code makes is to equidistribute the second spatial derivative of the initial condition since the error one makes by a piecewise linear approximation is proportional to that derivative.

The choice of initial mesh has an effect on the adaptive properties of this method. In order to study this effect we once again solved Eqs. (25) with the same values of TOL and initial values of N and Δt. Instead of an initially uniform mesh, however, we allowed the code to determine an equidistributed one. The results for TOL = 0.8 and TOL = 0.5 are presented in Tables 9 and 10, respectively. Mesh trajectories corresponding to the data presented in Tables 9 and 10 are shown in Figures 7 and 8, respectively (at the end of Example 2).

Table 9 **Numerical Parameters for Solving Eqs. (25) on an Initially Equidistributed Mesh with TOL = 0.8 and N and Δt Initially 10 and 0.1, Respectively**

Significant Times	N (10)	Δt (0.1)	Error Estimate
0.0541	-	0.0541	0.4867
0.0860	-	0.0320	0.5449
0.1121	-	0.0260	0.4995
0.1381	22	-	0.5145
0.3465	-	-	0.5449
0.3725	45	-	0.4480
1.0000	-	-	0.5087

Table 10 **Numerical Parameters for Solving Eqs. (25) on an Initially Equidistributed Mesh with TOL = 0.5 and N and Δt Initially 10 and 0.1, Respectively**

Significant Times	N (10)	Δt (0.1)	Error Estimate
0.0428	19	0.0428	0.3425
0.0643	-	0.0187	0.3060
0.3416	-	-	0.4149
0.3603	49	-	0.2909
0.7898	-	-	0.4630
0.8085	70	-	0.1283
1.0000	-	-	0.2481

These results are very interesting. First, compare the results for TOL equal to 0.8 (cf., Tables 7 and 9). In this case, it is not clear which initial mesh strategy is best. One case uses slightly less elements in general (cf., Table 9), while the other uses slightly larger time steps (cf., Table 7). One case only performs spatial and temporal refinements independently (cf., Table 9), while the other performs two full refinements (cf., Table 7) but performs fewer refinement steps

overall. It seems that for such a crude tolerance, an initially equidistributed mesh gains a user no real advantage and perhaps in some ways hinders the process.

On the other hand, comparing the results for TOL equal to 0.5 leads to a completely different conclusion (cf., Tables 8 and 10). In this case, starting the calculations with an equidistributed mesh is clearly advantageous. With such an initial mesh, the code uses far fewer elements (70 versus 103 ultimately) and performs fewer refinement steps overall (4 versus 6). Even though this strategy produces a smaller ultimate time step (0.0187 versus 0.0193), the difference is not great. Also, this strategy does take a larger initial time step (0.0428 versus 0.0394) and determines its ultimate time step more quickly (one less refinement step).

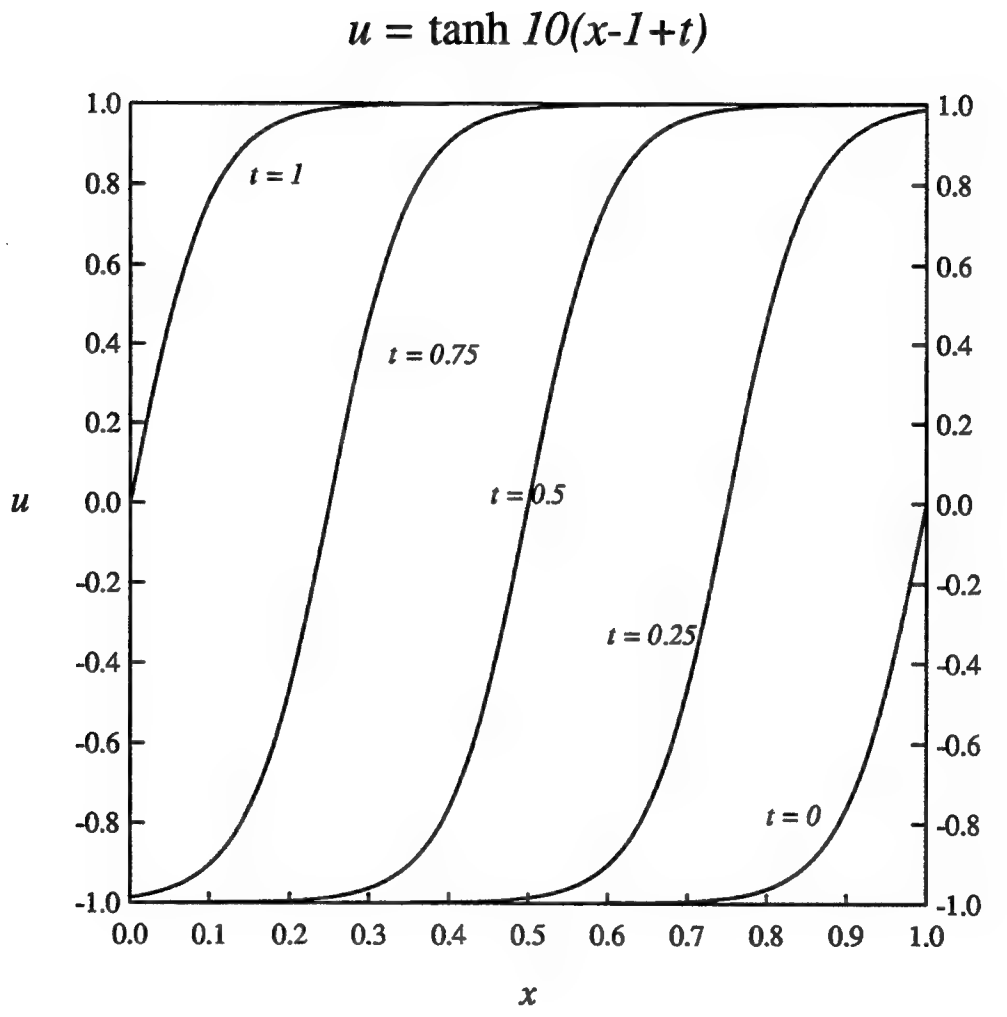


Figure 4. Graphs of Eq. (25b) for selected instances of time.

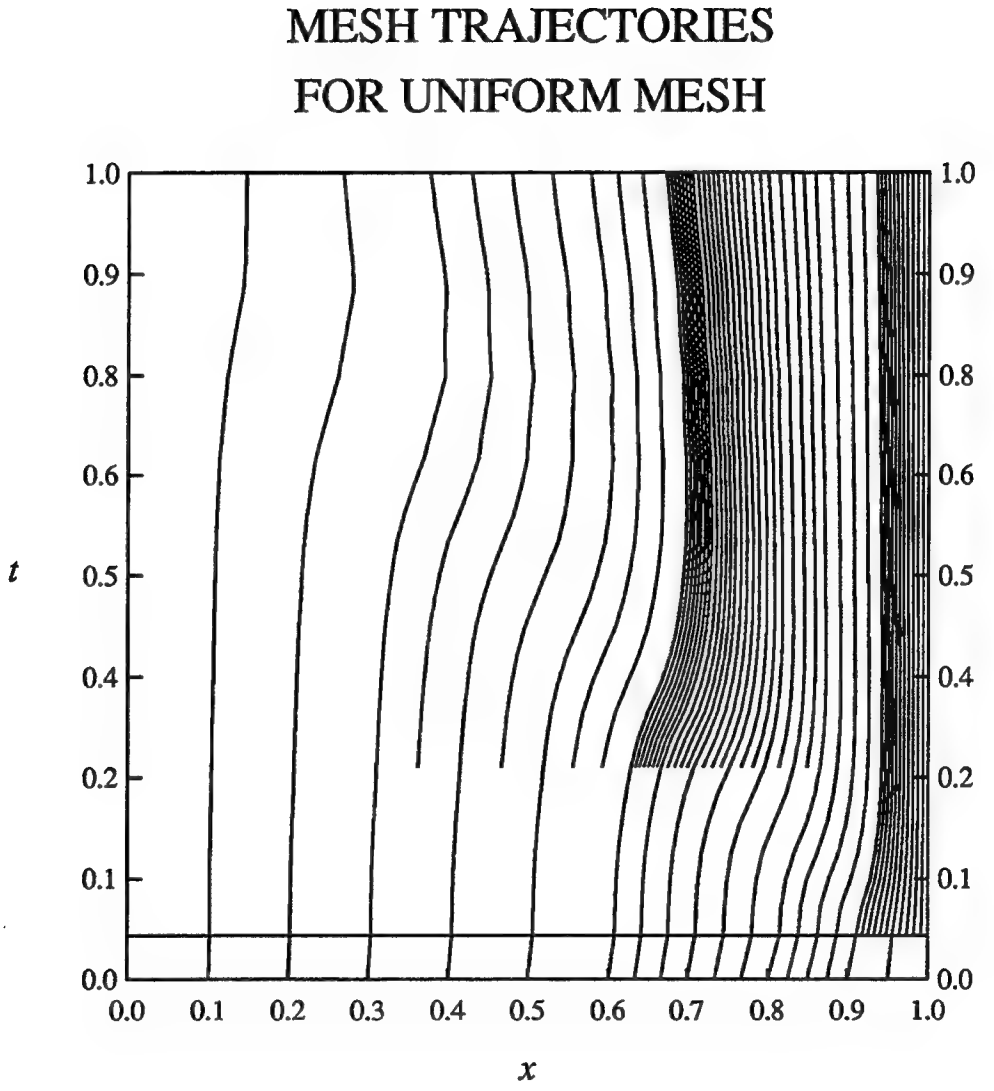


Figure 5. Mesh trajectories for Example 2 corresponding to Table 7.

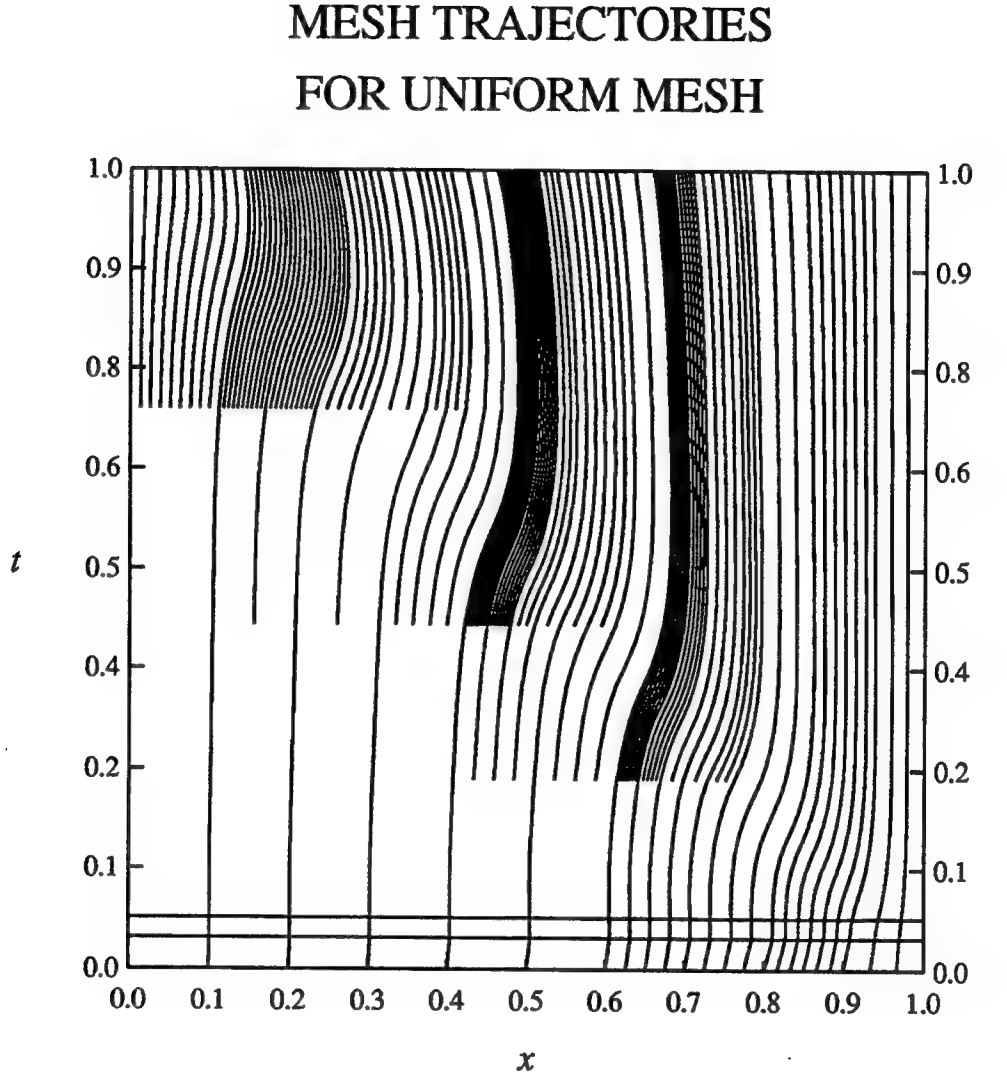


Figure 6. Mesh trajectories for Example 2 corresponding to Table 8.

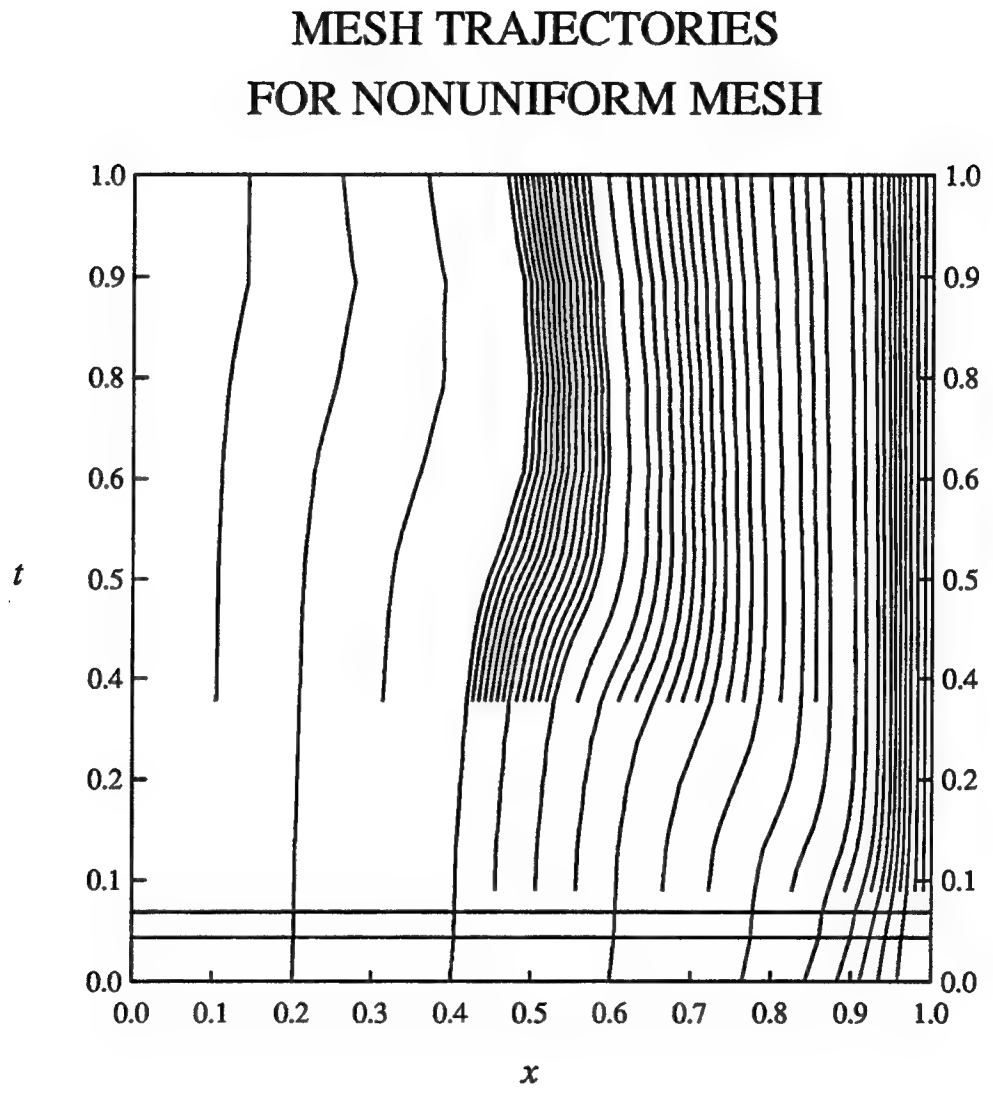


Figure 7. Mesh trajectories for Example 2 corresponding to Table 9.

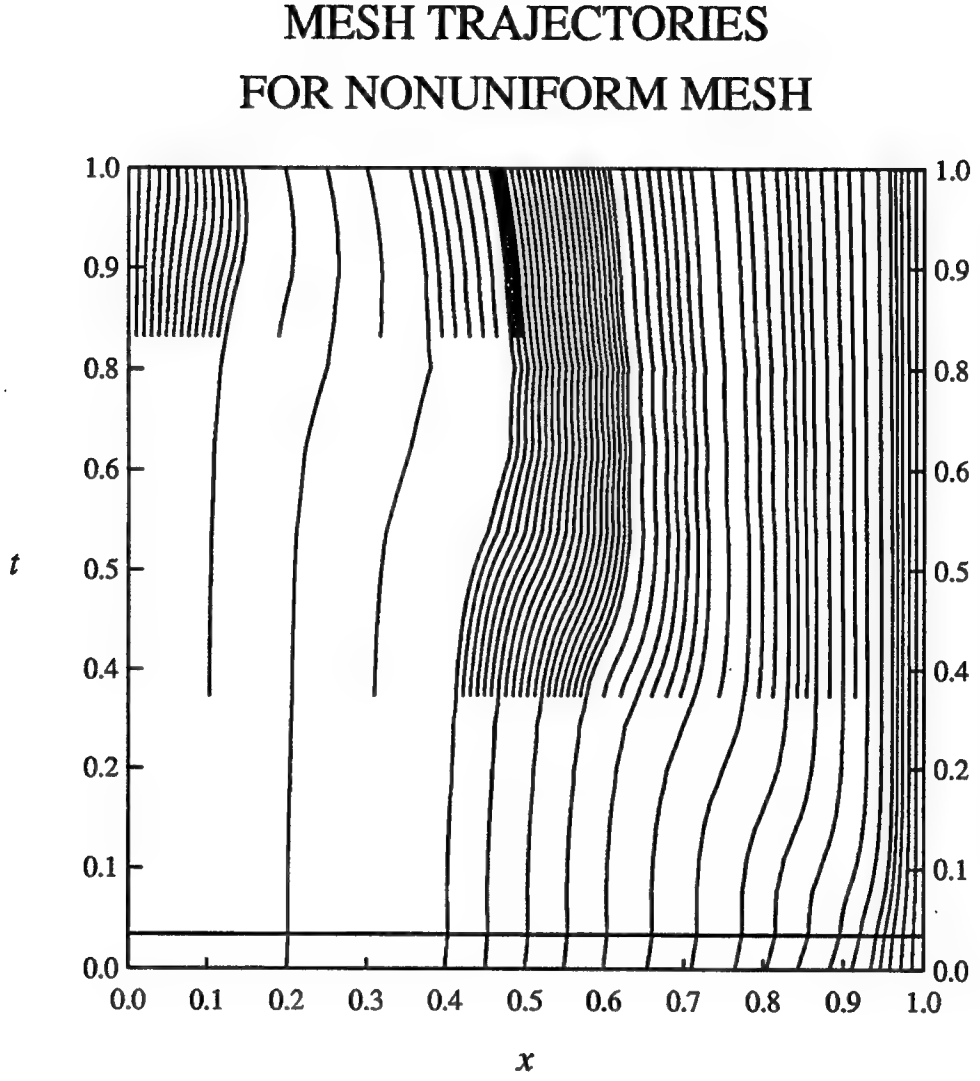


Figure 8. Mesh trajectories for Example 2 corresponding to Table 10.

Example 3

Consider the same partial differential equation as in Example 2:

$$u_t + f(x,t,u,u_x) = \frac{1}{2r_1^2} u_{xx},\ 0 < x < 1,\ t > 0. \tag{26a}$$

This time however, the initial and boundary conditions and the function f are chosen such that

$$u(x,t) = \tanh r_1[(x-1) + r_2 t] - \tanh r_1[x - r_2 t] \tag{26b}$$

is the exact solution. In addition to a wave that travels in the negative x direction when r_1 and r_2 are positive, function (26b) is composed of a wave that travels in the positive x direction (cf., Figure 9 at the end of Example 3). We solved this problem for values of r_1 and r_2 equal to 10 and 1, respectively, in order to study the effect of more than one moving structure on our adaptive algorithms.

There is another facet of the code that we decided to test with this particular problem. This facet is related to the dynamic equidistribution law upon which our moving algorithm is based (cf., the previous section and Coyle and Flaherty (ref 43)). Formally stated, the equidistribution problem is to determine a dynamic mesh

$$\Pi_N(t) = \{a = x_0 < x_1(t) < x_2(t) < \ldots < x_{N-1}(t) < x_N = b\} \tag{26c}$$

so that

$$\int_a^{x_j(t)} w(\zeta,t)\, d\zeta = \frac{j}{N} \int_a^b w(\zeta,t)\, d\zeta,\ j = 1,2,\ldots,N,\ t \geq 0, \tag{26d}$$

where $w(x,t) \geq 0$, $x \in [a,b]$, $t \geq 0$, is usually chosen to be a function of the solution of the underlying partial differential equation (cf., (ref 43)). This equidistribution problem has a nonunique solution whenever $w(x,t) := 0$. This difficulty is usually handled by imposing a lower bound on w, e.g., it is common to replace $w(x,t)$ by $w(x,t) + \eta$. This positive parameter η not only guarantees a unique solution to the equidistribution problem, but also can exercise a control on the mesh movement.

Whenever η is small compared to w, then w will continue to control the mesh movement. However, if η is large, then w has little effect, the constant η controls the mesh movement, and hence, the movement is restricted since the solution to Eq. (26d) is no movement when w is a constant. Different values of η can be input into the code depending on how much movement the user wishes to allow.

The effect of the parameter η can be quantified. Let $x_j(t)$, $y_j(t)$, and u_j denote mesh trajectories for $t \geq 0$ such that

$$\int_a^{x_j(t)} [w(\zeta,t) + \eta]\, d\zeta \;=\; \frac{j}{N} \int_a^b [w(\zeta,t) + \eta]\, d\zeta\,, \tag{26e}$$

$$\int_a^{y_j(t)} w(\zeta,t)\, d\zeta \;=\; \frac{j}{N} \int_a^b w(\zeta,t)\, d\zeta\,, \tag{26f}$$

and

$$\int_a^{u_j} \eta\, d\zeta \;=\; \frac{j}{N} \int_a^b \eta\, d\zeta\,. \tag{26g}$$

This implies that

$$\int_a^{x_j(t)} [w(\zeta,t) + \eta]\, d\zeta \;=\; \int_a^{y_j(t)} w(\zeta,t)\, d\zeta + \int_a^{u_j} \eta\, d\zeta\,, \tag{26h}$$

which further implies

$$\int_{y_j(t)}^{x_j(t)} w(\zeta,t)\, d\zeta + \int_{u_j}^{x_j(t)} \eta\, d\zeta \;=\; 0\,. \tag{26i}$$

Invoking the Mean Value Theorem yields

$$x_j(t) \;=\; \frac{w(x(t),t)}{w(x(t),t) + \eta}\, y_j(t) \;+\; \frac{\eta}{w(x(t),t) + \eta}\, u_j\,, \tag{26j}$$

where $x(t)$ is some function whose values lie between $x_j(t)$ and $y_j(t)$. Hence, $x_j(t)$ is a linear combination of the moving trajectory $y_j(t)$ and the stationary trajectory u_j with the value of η determining the dominant component.

Furthermore, Eq. (26j) implies

$$|x_j(t) - u_j| \;\le\; \frac{\max\limits_{a \le x \le b,\, t \ge 0} w(x,t)}{\eta}\, |y_j(t) - u_j|\,. \tag{26k}$$

Choosing η to be proportional to the maximum value of w yields

$$|x_j(t) - u_j| \;\le\; \frac{1}{r}\, |y_j(t) - u_j| \tag{26l}$$

where

$$\eta = r \max_{a \le x \le b, t \ge 0} w(x,t) . \qquad \textbf{(26m)}$$

Two different values of the parameter, η, were selected to see how the adaptive algorithms reacted. An initial value of N equal to 20 and Δt equal to 0.05 were used in both instances with TOL = 0.5. Table 11 summarizes the results for a value of η equal to 10, while Table 12 presents results when η was set to 2. Both tables are similar to the tables presented in Example 2. Mesh trajectories corresponding to the data presented in Tables 11 and 12 are shown in Figures 10 and 11, respectively (at the end of Example 3). The horizontal lines in these figures are used to distinguish the regions where the values of Δt are different.

Table 11 **Numerical Parameters for Solving Eqs. (26) on an Initially Equidistributed Mesh with TOL = 0.5 and N and Δt Initially 20 and 0.05, Respectively, and η = 10.**

Significant Times	N (20)	Δt (0.05)	Error Estimate
0.0360	41	0.0360	0.2595
0.0655	-	0.0295	0.2945
0.0950	68	-	0.2841
0.1540	-	-	0.3268
0.1836	102	-	0.2929
0.3017	-	-	0.3507
0.3312	135	-	0.2759
1.0000	-	-	0.2025

Table 12 **Numerical Parameters for Solving Eqs. (26) on an Initially Equidistributed Mesh with TOL = 0.5 and N and Δt Initially 20 and 0.05, Respectively, and $\eta = 2$**

Significant Times	N (20)	Δt (0.05)	Error Estimate
0.0382	39	0.0382	0.2623
0.0764	-	-	0.3243
0.1147	71	-	0.2794
0.1911	-	-	0.3278
0.2233	112	0.0322	0.2659
0.3845	-	-	0.3512
0.4068	-	0.0223	0.2666
0.4515	-	-	0.3458
0.4654	-	0.0139	0.2505
1.0000	-	-	0.1032

The interpretation of these results is not as straightforward as the previous two examples. At first, when the movement is less restricted (cf., Table 12) the code is able to take larger time steps and use fewer elements. However, as the solution process continues, the smaller number of elements results in a smaller time step.

When the movement is more restricted (cf., Table 11), a reasonably refined time step is determined quickly, but more and more elements must be used in order to solve the problem within the specified tolerance. The decision on the restriction of movement would seem to depend on what is more costly for the individual problem being addressed. More restrictive movement may result in an ultimately larger time step but with, perhaps, a larger number of elements being used at each of those time levels. Is that really better than an ultimately smaller time step with far less elements and with a succession of intermediate time steps being used before the final temporal refinement? The answer would seem to depend on the magnitude of the numbers for the problem. If the number of elements that must be used when restricting the movement is so large that solving the problem at any time level is very expensive in terms of computer time and storage, then restricting the movement was not the way to proceed. On the other hand, if unrestricted movement causes time steps so small that this is the source of unreasonable computer times, then the movement should be more restricted.

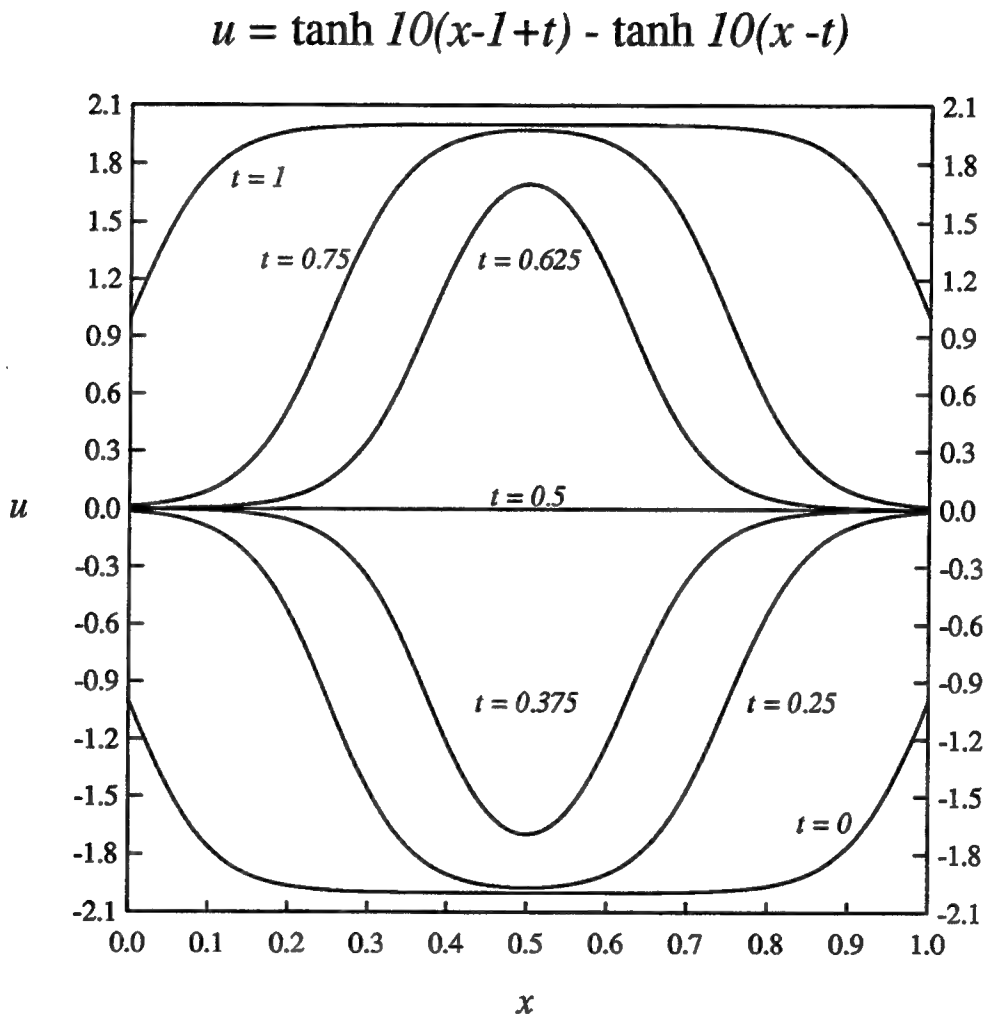


Figure 9. Graphs of Eq. (26b) for selected instances of time.

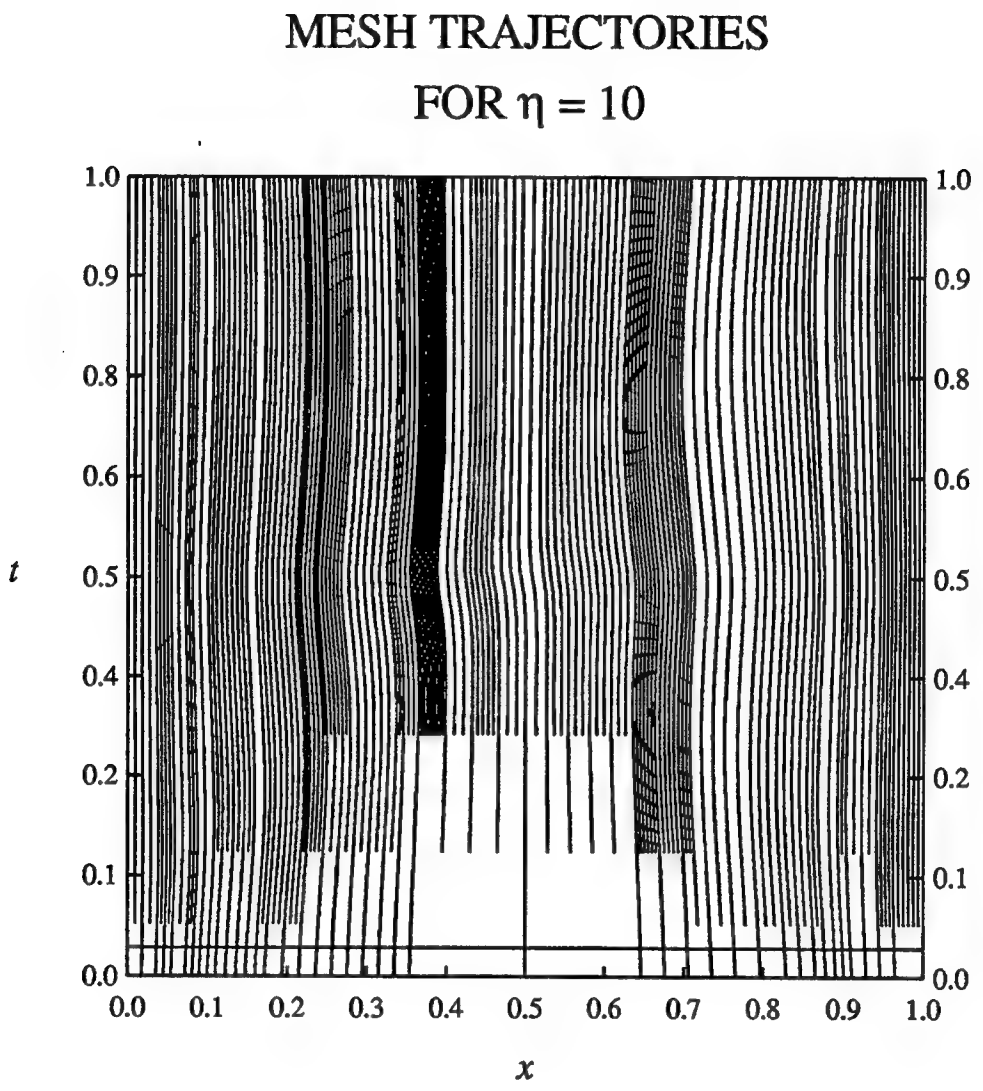


Figure 10. Mesh trajectories for Example 3 corresponding to Table 11.

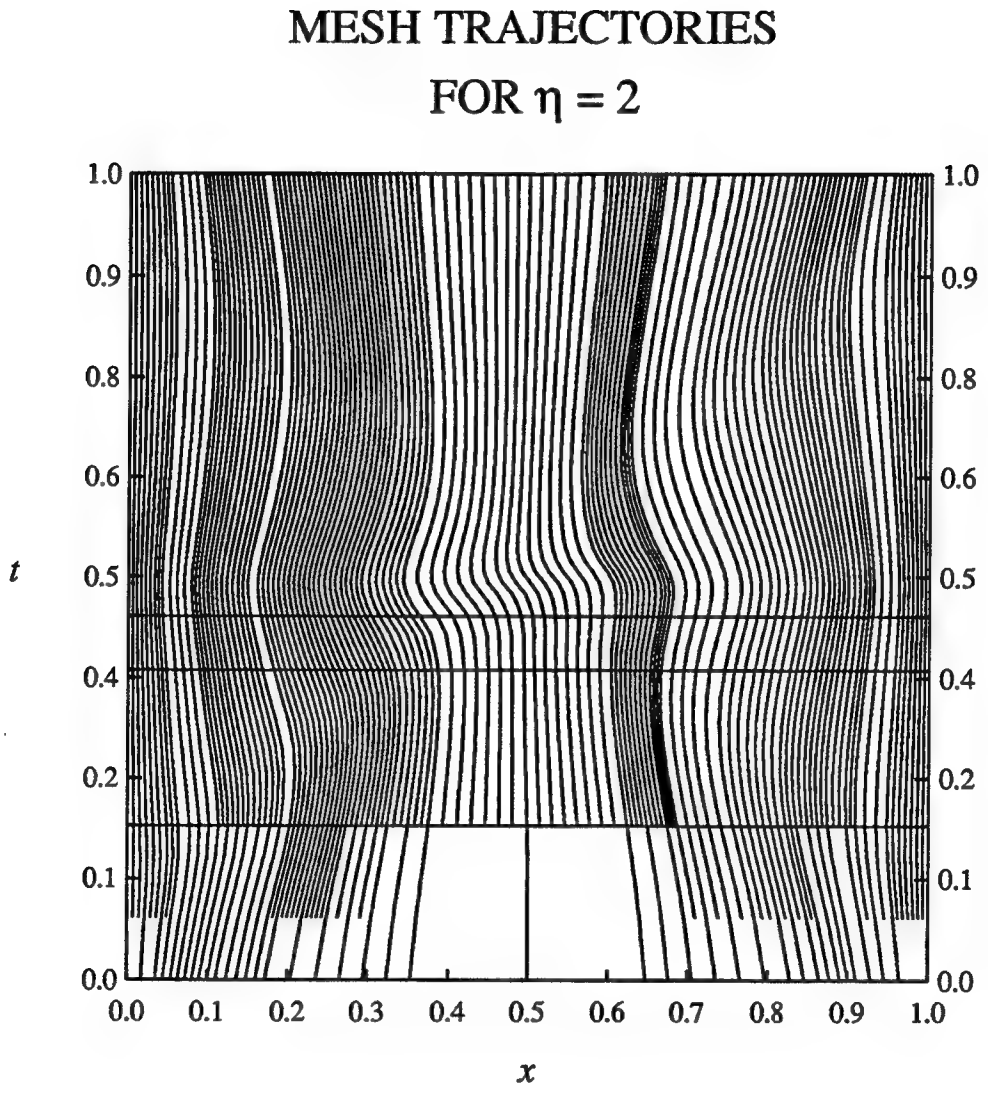


Figure 11. Mesh trajectories for Example 3 corresponding to Table 12.

Example 4

Consider an elastic impact problem for circular cylindrical rods as proposed by T. W. Wright (ref 46) which satisfies:

$$v = w_t,\ e = w_x,\ p = u_t,\ q = u_x, \qquad \textbf{(27a,b,c,d)}$$

$$e_t = v_x,\ q_t = p_x, \qquad \textbf{(27e,f)}$$

$$v_t = S_x,\ p_t = 2(Q_x - P). \qquad \textbf{(27g,h)}$$

Here, u and w are dimensionless radial and axial displacement components, p and v are radial and axial velocity components, e is the axial strain, q is the shear strain, and S, P, and Q are axial, radial, and shear forces, respectively. Equations (27e,f) are compatibility relations, and Eqs. (27g,h) are the equations of motion. We also need appropriate constitutive laws that relate S, P, and Q to e, u, and q, and herein we simply use the linear Hooke's laws

$$S = e + \frac{2\nu}{1-\nu}u,\ P = \frac{2}{1-\nu}(\nu e + u),\ Q = \frac{1-2\nu}{4(1-\nu)}q, \qquad \textbf{(27i,j,k)}$$

where ν is Poisson's ratio. If $\nu = 0$, we have a one-dimensional theory and Eqs. (27) reduce to a simple wave equation. However, if $\nu \neq 0$, Eqs. (27) give a two-dimensional theory that is valid when the length L of the cylindrical rod is large compared to its unit radius.

We use $\nu = 0.3$, homogeneous initial conditions and the boundary conditions

$$v(0,t) = 0.8919,\ Q(0,t) = 0,\ S(11,t) = 0,\ Q(11,t) = 0. \qquad \textbf{(27l,m,n,o)}$$

These conditions correspond to a cylinder of length $L = 11$ that is hit at $t = 0$ by a rigid wall that is moving with a velocity of 0.8919. We also added viscous terms ϵq_{xx}, ϵv_{xx}, and ϵp_{xx} to the right hand sides of Eqs. (27d,e,f), respectively, and used a value of $\epsilon = 0.01$.

Strictly speaking, this is not the type of problem that this method was designed to solve. Equations (27) are a system of hyperbolic equations not parabolic. However, the method has shown the ability to successfully solve problems even when the diffusion matrix, $D(x,t,u)$, of the model equations (cf., Eq. (3) is small (cf., Examples 1, 2, and 3). This lead to the addition of the diffusive terms as mentioned above and the determination of whether such diffusion terms would be enough to enable the code to solve such problems.

It turned out, however, that the introduction of artificial diffusion was not enough for this particular problem. The backward Euler method, our basic time integrator is a super stable method that tends to dissipate oscillations even when they are real. This would lead our code to conclude that the temporal error was large and hence cut the time step to unacceptable levels.

To avoid this dilemma, we did not propagate the backward Euler solution when solving this problem. Instead, we propagated the trapezoidal rule solution and used the backward Euler method to furnish the local error estimates. This is not quite correct because the backward Euler solution is of a lower order than the trapezoidal rule one. However, other researchers have obtained error estimates in this manner and have named the procedure "reverse" or "local extrapolation" (cf., Hairer, Norsett, and Wanner (ref 47)). It seemed a reasonable way to proceed given our framework.

This problem was proposed to us by a colleague (cf., Flaherty, Coyle, Ludwig, and Davis (ref 48)). He was mainly interested in solving for the axial stress and comparing the numerical results with his theoretical predictions. He had no real interest in the other stress or displacement components.

This code gives the user the ability to concentrate on some subset of the components of the vector system of equations and ignore others. This is done by having the user base adaptivity on a subset of the equations.

When we solved this problem, the code was instructed to base its adaptivity only on the equation for the axial stress. Hence, when the tolerance parameter, TOL, was set to 0.5, it was only the axial stress that was determined to within that tolerance. Since the axial stress was coupled to the other components, we wished to study the effect of selecting only one component on which to base our adaptive algorithms.

The results of solving this problem are very encouraging. The refinement and error estimation procedures detected the discontinuity at $x = 0, t = 0$ and immediately concentrated more elements there as well as cut the time step (cf., Figure 12 at the end of Example 4).

Initially, N was 50 and Δt was 0.01 and the code changed this to $N = 75$ and $\Delta t = 0.0051$. The time step was the changed to 0.0017 and no refinement was necessary after that. Experience with this problem has shown that no additional refinement was necessary more as a result of the level of the immediate refinement than of the mesh moving. This is probably due to the difficulty associated with the initial discontinuity.

The axial stress was more than adequately determined (cf., Figure 13 at the end of Example 4), as hoped, when compared to theoretical predictions and solutions computed with a uniformly fine discretization ($N = 400$, $\Delta t = 0.0005$) (cf., Figure 14 at the end of Example 4). It is known that one-dimensional elastic waves are nondispersive, whereas two-dimensional waves are dispersive. The dispersive nature of the two-dimensional solution is clearly visible in Figure 13.

Other components were not approximated as well. Occasional oscillations would develop (cf., Figure 15 at the end of Example 4). However, the qualitative nature of these solutions remained intact when compared to solutions determined using a uniformly fine discretization (cf., Figure 16 at the end of Example 4). Therefore, if a user is only interested in qualitative information on some problem components, it seems possible to save some computer effort by instructing the adaptive algorithms to ignore or reduce the accuracy of those components.

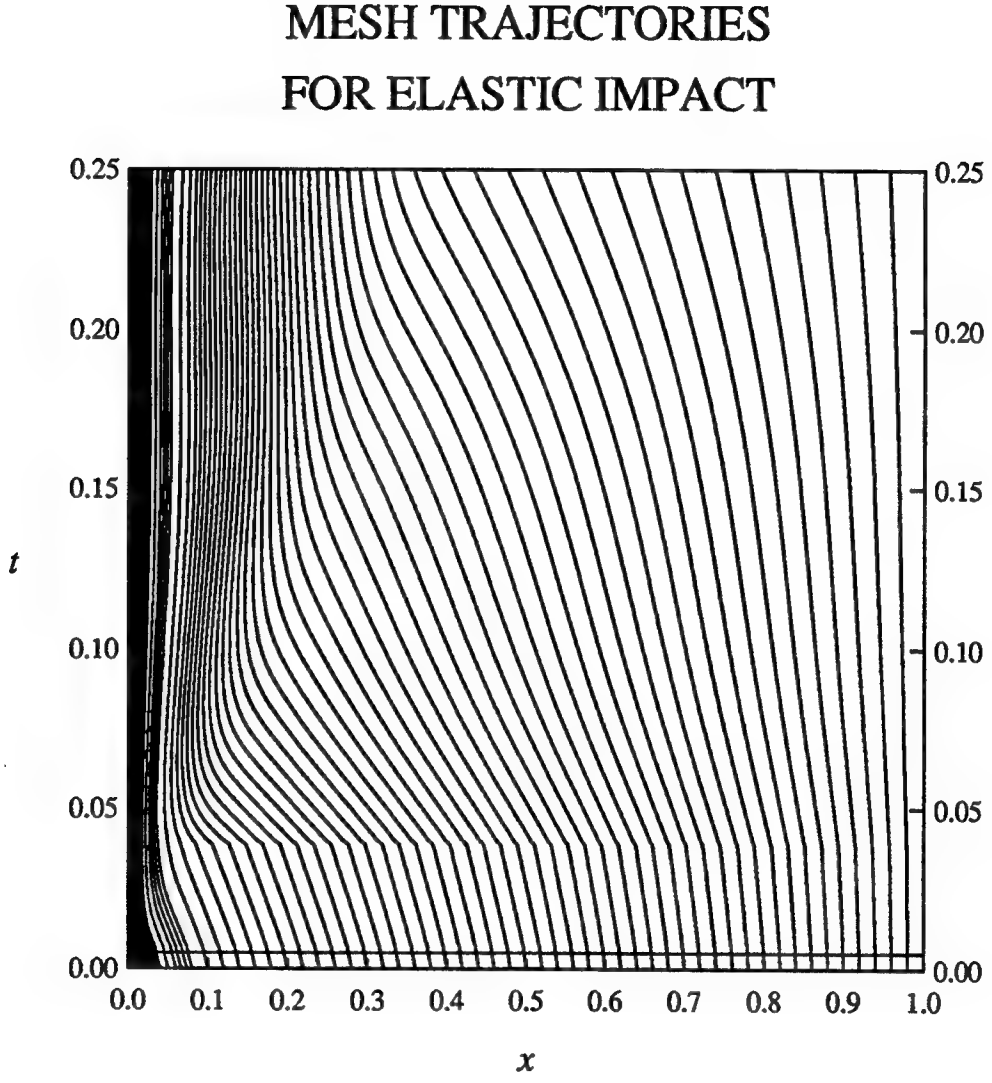


Figure 12. Mesh trajectories for Example 4.

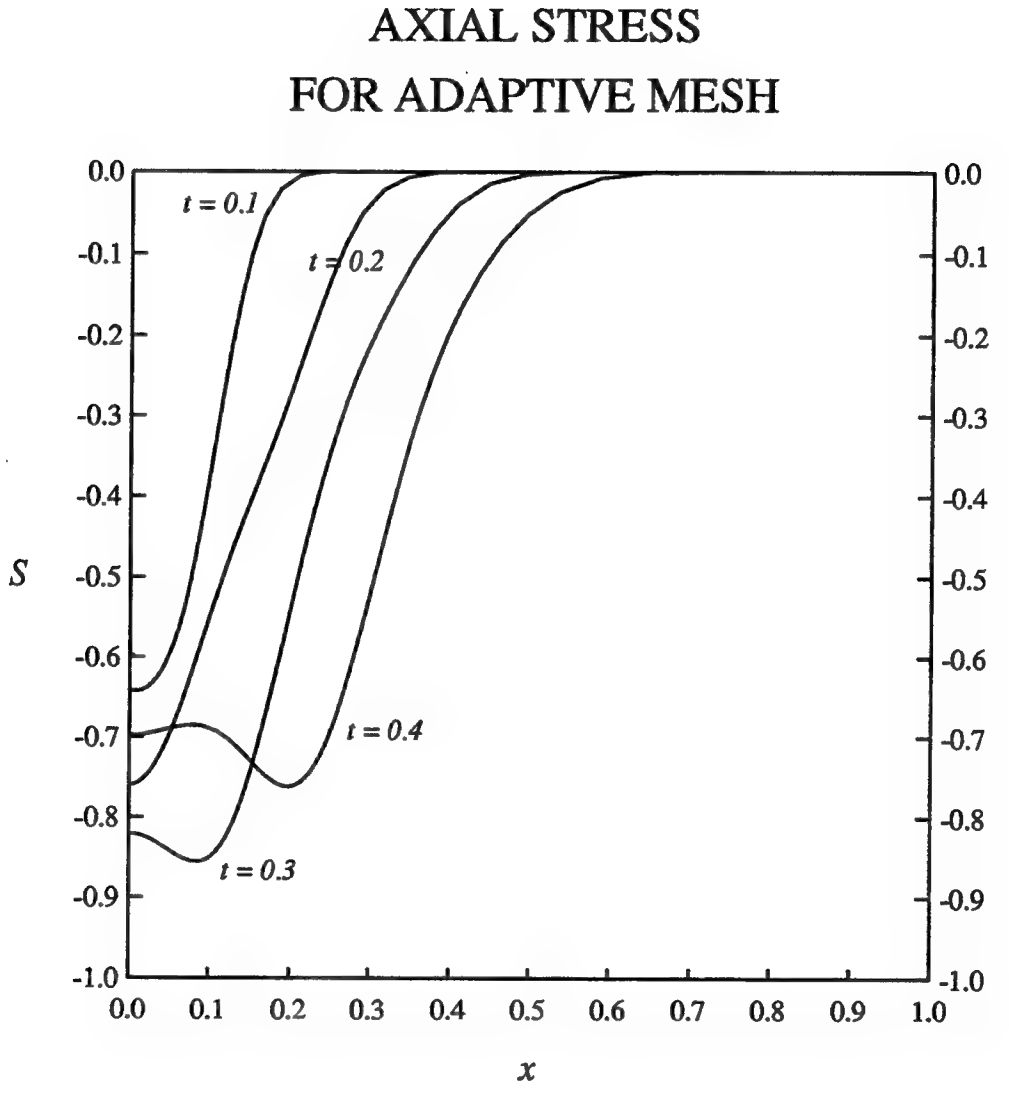


Figure 13. Axial stress versus position for four instances of time.

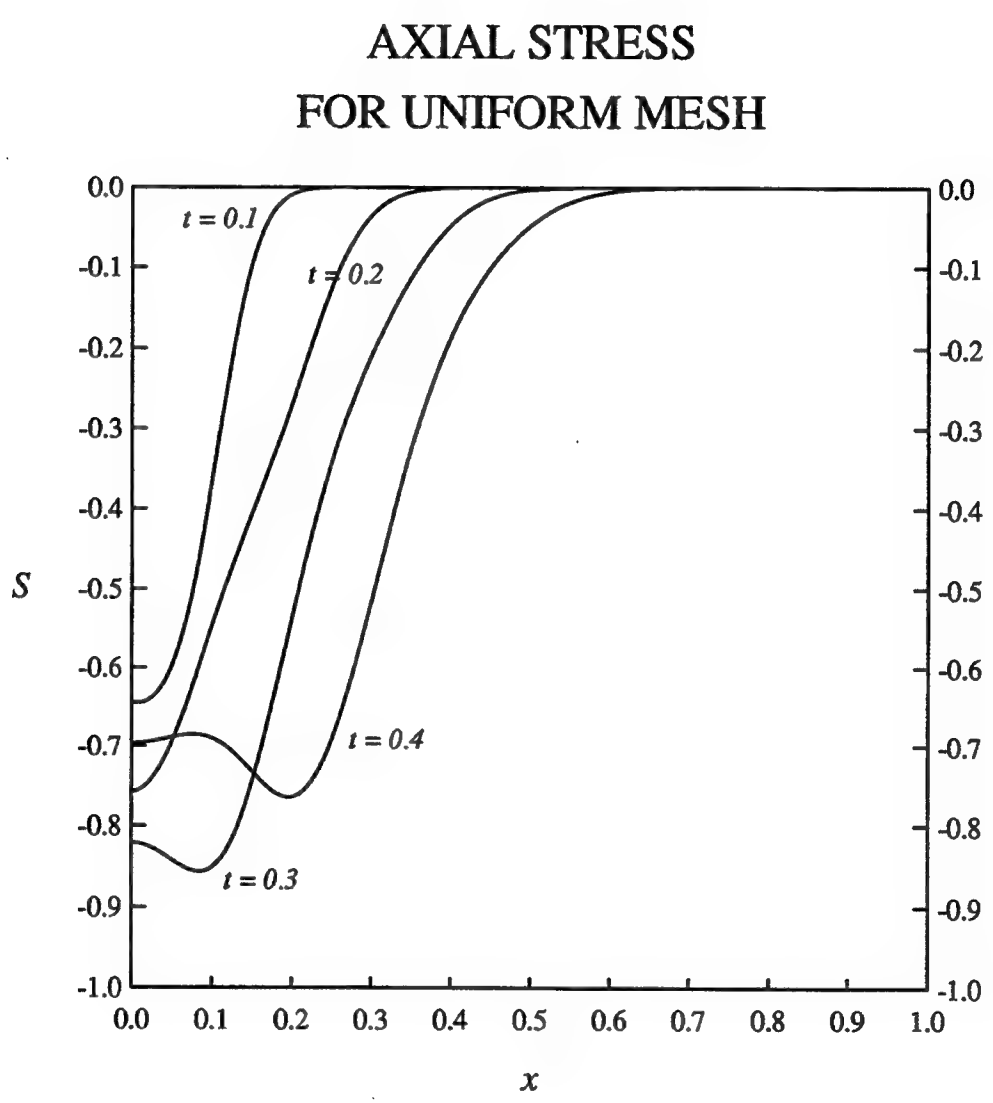


Figure 14. Axial stress versus position for four instances of time.

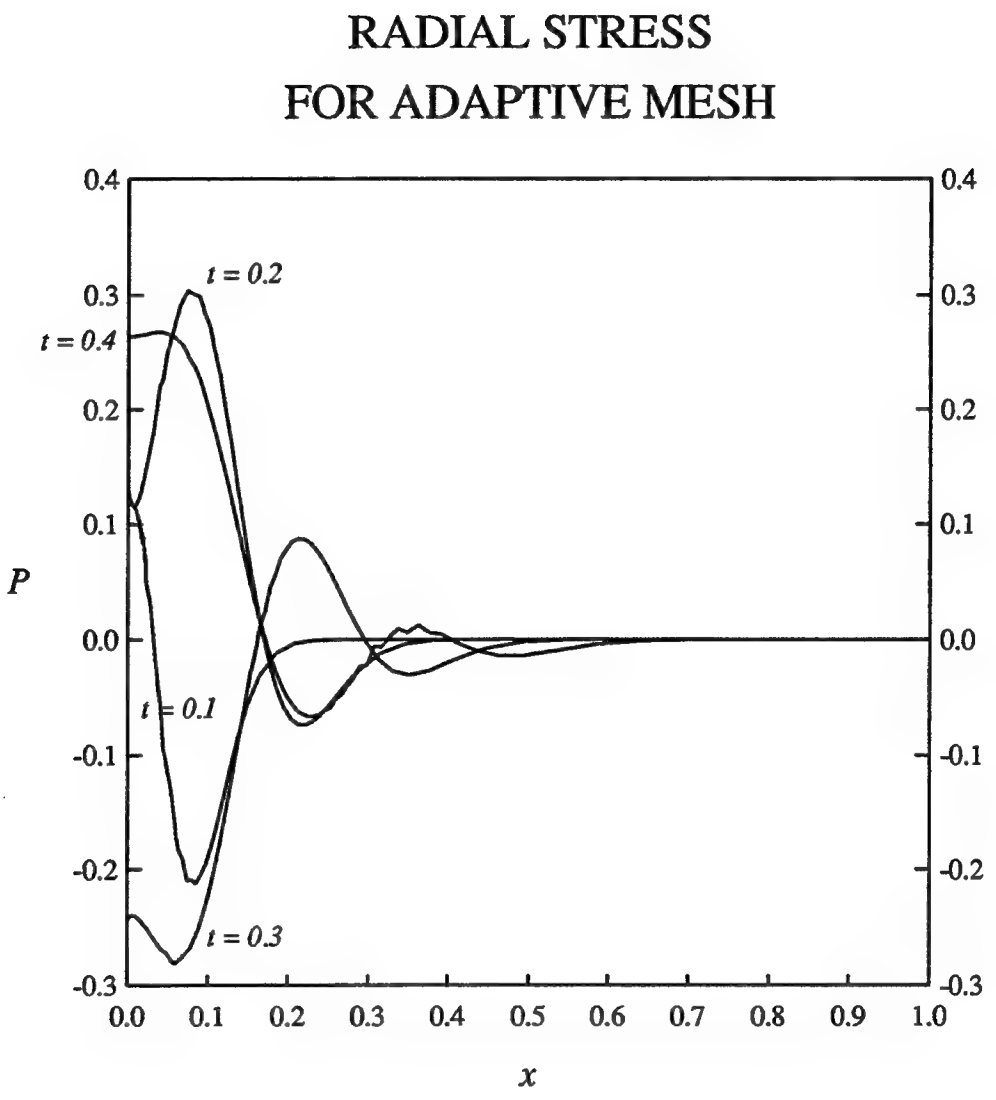


Figure 15. Radial stress versus position for four instances of time.

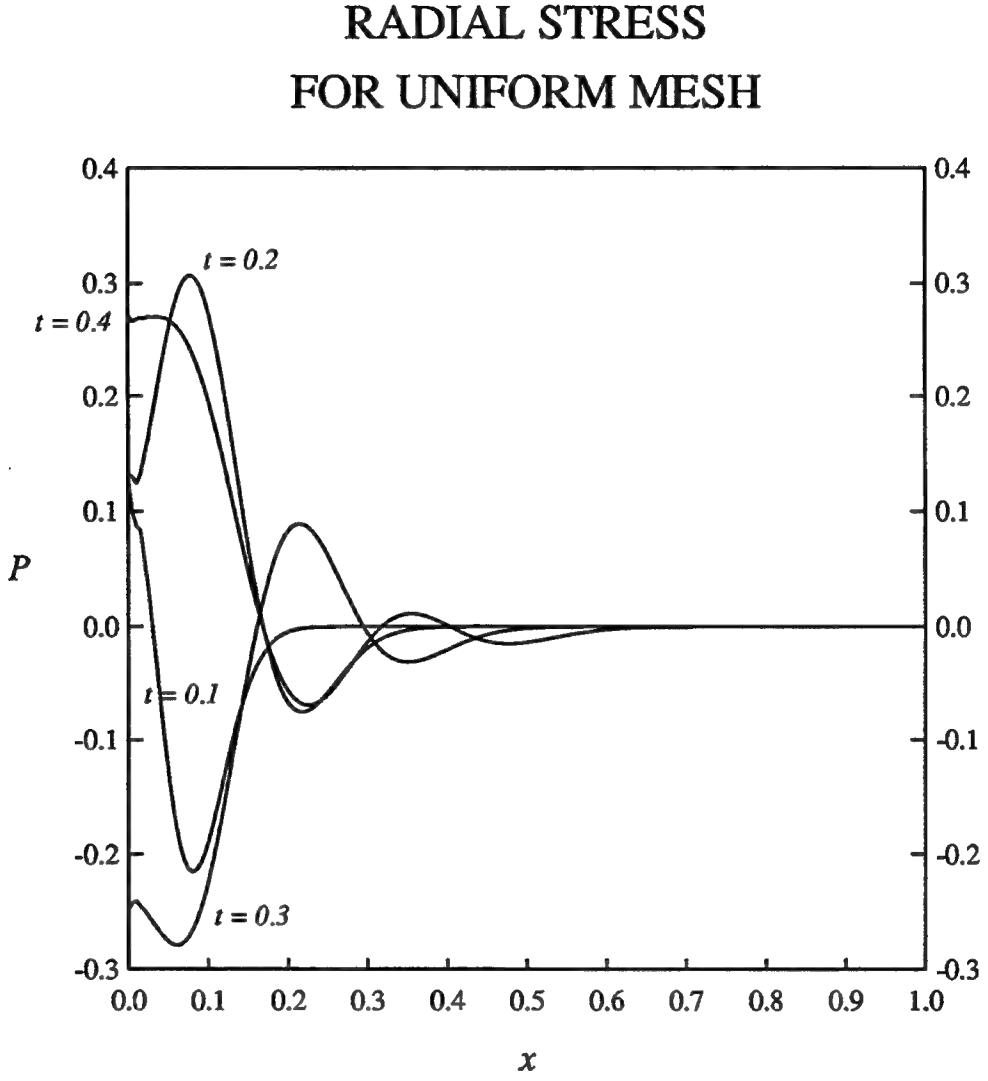


Figure 16. Radial stress versus position for four instances of time.

SUMMARY

This is the first in a series of reports whose overall purpose is to report on the progress of a project whose aim was to develop a reliable and efficient numerical method for solving systems of parabolic partial differential equations. In particular, the goal was for the resulting software to be able to compute an approximate solution within a user specified error tolerance while using a minimum of computer resources.

The basic approach taken to accomplish this goal was to combine the adaptive techniques of mesh movement and local mesh refinement. The reasons for this choice were that mesh movement could calculate the most reliable solution at a fixed discretization level, while local mesh refinement could guarantee an accurate solution for any prescribed error tolerance. Local mesh refinement schemes can be costly, due to the necessity of recomputing the solution; however, proper mesh motion should permit as few levels of refinement as possible (cf., Examples section).

The Problem Statement section formally stated the problem that AFEM attempts to solve. Then, in the Spatial Discretization section, the discretization for both piecewise linear and piecewise quadratic elements, defined on a moving mesh, was presented. The discretization for two time integration schemes was demonstrated in the Temporal Discretization section.

Although the finite element method is now fairly common, the concept of adaptive finite elements is still relatively new. An auxiliary purpose of this particular report was to not only give a brief description of the adaptive and solution algorithms developed for use in AFEM, but also to provide an introduction to adaptive strategies in general. This was done in the Adaptive and Solution Algorithms section.

In the Examples section, examples were presented so as to demonstrate the workings of the entire adaptive finite element code and how the different pieces interact. The results indicate that we have satisfied our original objective to a large degree. Solutions are computed to within a prescribed error tolerance with a much coarser level of discretization than would be necessary without adaptivity. The mesh movement does, indeed, delay the necessity for refinement. Meanwhile, refinement does not interfere with the stability and smoothness of the mesh motion.

In the future, we intend to extend the method to higher spatial dimensions. The question of proper and stable mesh movement has retarded our progress in this important area. With this obstacle now overcome (cf., Coyle and Flaherty (ref 43)), we feel we can more than adequately address this topic.

REFERENCES

1. C.W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1971.

2. L.R. Petzold, "A Description of DASSL: A Differential/Algebraic System Solver," Technical Report SAND 82-8637, Sandia National Laboratory, Livermore, CA, 1982.

3. I. Babuska, J. Chandra, and J.E. Flaherty, Eds., *Adaptive Computational Methods for Partial Differential Equations*, Society of Industrial and Applied Mathematicians, Philadelphia, 1983.

4. J.E. Flaherty, P.J. Paslow, M.S. Shephard, and J.D. Vasilakis, Eds., *Adaptive Methods for Partial Differential Equations*, Society of Industrial and Applied Mathematicians, Philadelphia, 1988.

5. L. Lapidus and G.F. Pinder, *Numerical Solution of Partial Differential Equations in Science and Engineering*, John Wiley and Sons, New York, 1982.

6. A. Friedman, "The Stefan Problem in Several Space Variables," *Trans. Amer. Math. Soc.*, Vol. 133, 1968, pp. 51-87.

7. E.F. Keller and G.M. Odell, "Necessary and Sufficient Conditions for Chemotactic Bands," *Math. Biosci.*, Vol. 27, 1975, pp. 309-317.

8. A.K. Kapila, "Reactive-Diffusive Systems with Arrhenius Kinetics: Dynamics of Ignition," *SIAM J. Appl. Math.*, Vol. 39, 1980, pp. 21-36.

9. P.C. Fife, "Singular Perturbation and Wave Front Techniques in Reaction-Diffusion Problems," *SIAM-AMS Proceedings*, Vol. 10, 1975, pp. 23-49.

10. F. Hoppensteadt, "Mathematical Theories of Population: Demographies, Genetics, and Epidemics," *Regional Conference Series in Applied Math.*, Vol. 20, SIAM, Philadelphia, 1975.

11. G.K. Batchelor, *An Introduction to Fluid Dynamics*, Cambridge University Press, Cambridge, 1970.

12. S. Adjerid and J.E. Flaherty, "A Moving Finite Element Method with Error Estimation and Refinement for One-Dimensional Time-Dependent Partial Differential Equations," *SIAM J. Numer. Anal.*, Vol. 23, 1986, pp. 778-796.

13. I. Babuska and M.R. Dorr, "Error Estimates for the Combined h and p Versions of the Finite Element Method," *Numer. Math.*, Vol. 37, 1981, pp. 257-277.

14. I. Babuska, O.C. Zienkiewicz, J.R. Gago, and E.R. De Arantes E Oliveira, Eds., *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, John Wiley and Sons, London, 1986.

15. R.E. Bank and A. Weiser, "Some A Posteriori Error Estimates for Elliptic Partial Differential Equations," *Math. Comp.*, Vol. 44, 1985, pp. 283-301.

16. M.J. Berger and J. Oliger, "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations," *J. Comp. Phys.*, Vol. 53, 1984, pp. 484-512.

17. M. Bieterman and I. Babuska, "The Finite Element Method for Parabolic Equations, I. A Posteriori Error Estimation," *Numer. Math.*, Vol. 40, 1982, pp. 339-371.

18. M. Bieterman and I. Babuska, "The Finite Element Method for Parabolic Equations, II. A Posteriori Error Estimation and Adaptive Approach," *Numer. Math.*, Vol. 40, 1982, pp. 373-406.

19. P.K. Moore and J.E. Flaherty, "A Local Refinement Finite Element Method for Time-Dependent Partial Differential Equations," in: *Trans. Second Army Conf. on Appl. Math. and Comput.*, ARO Report 85-1, U.S. Army Research Office, Research Triangle Park, NC, 1985, pp. 585-595.

20. M.S. Shephard, N. Qingxiang, and P.L. Baehmann, "Some Results Using Stress Projectors for Error Indication and Estimation," in: *Adaptive Methods for Partial Differential Equations*, J.E. Flaherty, P.J. Paslow, M.S. Shephard, and J.D. Vasilakis, Eds., Society of Industrial and Applied Mathematicians, Philadelphia, 1988, pp. 83-99.

21. T. Strouboulis and J.T. Oden, "A Posteriori Estimation of the Error in Finite Element Approximations of Convection Dominated Problems," in: *Finite Element Analysis in Fluids: Proceedings of the Seventh International Conference on Finite Element Methods in Flow Problems*, T.J. Chung and G.R. Karr, Eds., University of Alabama in Huntsville Press, Huntsville, 1989, pp. 125-136.

22. O.C. Zienkiewicz and J.Z. Zhu, "A Simple Error Estimator and Adaptive Procedure for Practical Engineering Analysis," *Int. J. Num. Meth. Eng.*, Vol. 24, 1987, pp. 337-357.

23. D.C. Arney and J.E. Flaherty, "A Two-Dimensional Mesh Moving Technique for Time-Dependent Partial Differential Equations," *J. Comput. Phys.*, Vol. 67, 1986, pp. 124-144.

24. J.B. Bell and G.R. Shubin, "An Adaptive Grid Finite Difference Method for Conservation Laws," *J. Comput. Phys.*, Vol. 52, 1983, pp. 569-591.

25. S.F. Davis and J.E. Flaherty, "An Adaptive Finite Element Method for Initial-Boundary Value Problems for Partial Differential Equations," *SIAM J. Sci. Statist. Comput.*, Vol. 3, 1982, pp. 6-27.

26. E.A. Dorfi and L. O'C. Drury, "Simple Adaptive Grids for 1-D Initial Value Problems," *J. Comput. Phys.*, Vol. 69, 1987, pp. 175-195.

27. H.A. Dwyer, "Grid Adaptation for Problems with Separation, Cell Reynolds Number, Shock-Boundary Layer Interaction, and Accuracy," *AIAA Paper No. 83-0449*, AIAA Twenty-First Aerospace Sciences Meeting, 1983.

28. R.E. Ewing, T.F. Russell, and M.F. Wheeler, "Convergence Analysis of an Approximation of Miscible Displacement in Porous Media by Mixed Finite Elements and a Modified Method of Characteristics," *Computer Meth. Appl. Mech. Eng.* Vol. 47, 1984, pp. 161-176.

29. J.M. Hyman, "Adaptive Moving Mesh Methods for Partial Differential Equations," Los Alamos National Laboratory Report LA-UR-82-3690, Los Alamos National Laboratory, Los Alamos, NM, 1982.

30. E.J. Kansa, D.L. Morgan, and L.K. Morris, "A Simplified Moving Finite Difference Scheme: Application to Dense Gas Dispersion," *SIAM J. Sci. Stat. Comput.*, Vol. 5, 1984, pp. 667-683.

31. K. Miller and R. Miller, "Moving Finite Elements, Part I," *SIAM J. Num. Anal.*, Vol. 18, 1981, pp. 1019-1032.

32. K. Miller, "Moving Finite Elements, Part II," *SIAM J. Num. Anal.*, Vol. 18, 1981, pp. 1033-1057.

33. L.R. Petzold, "An Adaptive Moving Grid Method for One-Dimensional Systems of PDEs and its Numerical Solution in Adaptive Methods for Partial Differential Equations," in: *Adaptive Methods for Partial Differential Equations*, J. E. Flaherty, P. J. Paslow, M. S. Shephard, and J. D. Vasilakis, Eds., Society of Industrial and Applied Mathematicians, Philadelphia, 1988, pp. 253-265.

34. M.M. Rai and D.A. Anderson, "The Use of Adaptive Grids in Conjunction with Shock Capturing Methods," AIAA Paper No. 81-10112, June 1981.

35. R.D. Russell and Y. Ren, "Moving Mesh Techniques Based Upon Equidistribution and Their Stability," *SIAM J. Sci. Stat. Comput.*, Vol. 13, No. 6, 1992, pp. 1265-1286.

36. J.S. Saltzman and J.U. Brackbill, "Adaptive Rezoning for Singular Problems in Two Dimensions," *J. Comput. Phys.*, Vol. 46, 1982, pp. 342-368.

37. M.D. Smooke and M.L. Koszykowski, "Fully Adaptive Solutions of One-Dimensional Mixed Initial-Boundary Problems in Combustion," Technical Report SAND 83-8219, Sandia National Laboratory, Livermore, CA, 1983.

38. J.F. Thompson, "Grid Generation in Computational Fluid Dynamics," *AIAA Journal*, Vol. 22, 1984, pp. 1505-1523.

39. J.G. Verwer, J.G. Blom, R.M. Furzeland, and P.A. Zegeling, "A Moving Grid Method for One-Dimensional PDEs Based on the Method of Lines," in: *Adaptive Methods for Partial Differential Equations*, J.E. Flaherty, P.J. Paslow, M. S. Shephard, and J.D. Vasilakis, Eds., Society of Industrial and Applied Mathematicians, Philadelphia, 1988, pp. 160-175.

40. A.B. White, "On the Numerical Solution of Initial/Boundary Value Problems in One Space Dimension," *SIAM J. Numer. Anal.*, Vol. 23, 1982, pp. 683-697.

41. J.M. Coyle and J.E. Flaherty, "Adaptive Finite Element Method II: Error Estimation," U.S. Army ARDEC Technical Report, to be published.

42. J.M. Coyle and J.E. Flaherty, "Adaptive Finite Element Method III: Mesh Refinement," U.S. Army ARDEC Technical Report, to be published.

43. J.M. Coyle and J.E. Flaherty, "Adaptive Finite Element Method IV: Mesh Movement," U.S. Army ARDEC Technical Report, to be published.

44. C. De Boor, *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.

45. J.M. Coyle, J.E. Flaherty, and R. Ludwig, "On the Stability of Mesh Equidistribution Strategies for Time-Dependent Partial Differential Equations," *J. Comput. Phys.*, Vol. 62, 1986, pp. 26-39.

46. T.W. Wright, "Nonlinear Waves in Rods," Technical Report ARBRL-TR-02324, U. S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD, 1981.

47. E. Hairer, F.P. Norsett, and G.P. Wanner, *Solving Ordinary Differential Equations I.: Nonstiff Problems*, Springer-Verlag, Berlin, 1987.

48. J.E. Flaherty, J.M. Coyle, R. Ludwig, and S.F. Davis, "Adaptive Finite Element Methods for Parabolic Partial Differential Equations in Adaptive Computational Methods for Partial Differential Equations," in: *Adaptive Computational Methods for Partial Differential Equations*, I. Babuska, J. Chandra, and J. E. Flaherty, Eds., Society for Industrial and Applied Mathematicians, Philadelphia, 1983, pp. 144-164.

TECHNICAL REPORT INTERNAL DISTRIBUTION LIST

	NO. OF COPIES
CHIEF, DEVELOPMENT ENGINEERING DIVISION	
ATTN: SMCAR-CCB-DA	1
-DC	1
-DI	1
-DR	1
-DS (SYSTEMS)	1
CHIEF, ENGINEERING DIVISION	
ATTN: SMCAR-CCB-S	1
-SD	1
-SE	1
CHIEF, RESEARCH DIVISION	
ATTN: SMCAR-CCB-R	2
-RA	1
-RE	1
-RM	1
-RP	1
-RT	1
TECHNICAL LIBRARY	
ATTN: SMCAR-CCB-TL	5
TECHNICAL PUBLICATIONS & EDITING SECTION	
ATTN: SMCAR-CCB-TL	3
OPERATIONS DIRECTORATE	
ATTN: SMCWV-ODP-P	1
DIRECTOR, PROCUREMENT & CONTRACTING DIRECTORATE	
ATTN: SMCWV-PP	1
DIRECTOR, PRODUCT ASSURANCE & TEST DIRECTORATE	
ATTN: SMCWV-QA	1

NOTE: PLEASE NOTIFY DIRECTOR, BENÉT LABORATORIES, ATTN: SMCAR-CCB-TL OF ADDRESS CHANGES.

TECHNICAL REPORT EXTERNAL DISTRIBUTION LIST

	NO. OF COPIES
ASST SEC OF THE ARMY RESEARCH AND DEVELOPMENT ATTN: DEPT FOR SCI AND TECH THE PENTAGON WASHINGTON, D.C. 20310-0103	1
ADMINISTRATOR DEFENSE TECHNICAL INFO CENTER ATTN: DTIC-FDAC CAMERON STATION ALEXANDRIA, VA 22304-6145	12
COMMANDER U.S. ARMY ARDEC ATTN: SMCAR-AEE	1
SMCAR-AES, BLDG. 321	1
SMCAR-AET-O, BLDG. 351N	1
SMCAR-FSA	1
SMCAR-FSM-E	1
SMCAR-FSS-D, BLDG. 94	1
SMCAR-IMI-I, (STINFO) BLDG. 59 PICATINNY ARSENAL, NJ 07806-5000	2
DIRECTOR U.S. ARMY RESEARCH LABORATORY ATTN: AMSRL-DD-T, BLDG. 305 ABERDEEN PROVING GROUND, MD 21005-5066	1
DIRECTOR U.S. ARMY RESEARCH LABORATORY ATTN: AMSRL-WT-PD (DR. B. BURNS) ABERDEEN PROVING GROUND, MD 21005-5066	1
DIRECTOR U.S. MATERIEL SYSTEMS ANALYSIS ACTV ATTN: AMXSY-MP ABERDEEN PROVING GROUND, MD 21005-5071	1
COMMANDER ROCK ISLAND ARSENAL ATTN: SMCRI-ENM ROCK ISLAND, IL 61299-5000	1
MIAC/CINDAS PURDUE UNIVERSITY P.O. BOX 2634 WEST LAFAYETTE, IN 47906	1
COMMANDER U.S. ARMY TANK-AUTMV R&D COMMAND ATTN: AMSTA-DDL (TECH LIBRARY) WARREN, MI 48397-5000	1
COMMANDER U.S. MILITARY ACADEMY ATTN: DEPARTMENT OF MECHANICS WEST POINT, NY 10966-1792	1
U.S. ARMY MISSILE COMMAND REDSTONE SCIENTIFIC INFO CENTER ATTN: DOCUMENTS SECTION, BLDG. 4484 REDSTONE ARSENAL, AL 35898-5241	2
COMMANDER U.S. ARMY FOREIGN SCI & TECH CENTER ATTN: DRXST-SD 220 7TH STREET, N.E. CHARLOTTESVILLE, VA 22901	1
COMMANDER U.S. ARMY LABCOM MATERIALS TECHNOLOGY LABORATORY ATTN: SLCMT-IML (TECH LIBRARY) WATERTOWN, MA 02172-0001	2
COMMANDER U.S. ARMY LABCOM, ISA ATTN: SLCIS-IM-TL 2800 POWER MILL ROAD ADELPHI, MD 20783-1145	1

NOTE: PLEASE NOTIFY COMMANDER, ARMAMENT RESEARCH, DEVELOPMENT, AND ENGINEERING CENTER, U.S. ARMY AMCCOM, ATTN: BENÉT LABORATORIES, SMCAR-CCB-TL, WATERVLIET, NY 12189-4050 OF ADDRESS CHANGES.

TECHNICAL REPORT EXTERNAL DISTRIBUTION LIST (CONT'D)

	NO. OF COPIES
COMMANDER U.S. ARMY RESEARCH OFFICE ATTN: CHIEF, IPO P.O. BOX 12211 RESEARCH TRIANGLE PARK, NC 27709-2211	1
DIRECTOR U.S. NAVAL RESEARCH LABORATORY ATTN: MATERIALS SCI & TECH DIV CODE 26-27 (DOC LIBRARY) WASHINGTON, D.C. 20375	1 1
COMMANDER AIR FORCE ARMAMENT LABORATORY ATTN: AFATL/MN EGLIN AFB, FL 32542-5434	1
COMMANDER AIR FORCE ARMAMENT LABORATORY ATTN: AFATL/MNF EGLIN AFB, FL 32542-5434	1

NOTE: PLEASE NOTIFY COMMANDER, ARMAMENT RESEARCH, DEVELOPMENT, AND ENGINEERING CENTER, U.S. ARMY AMCCOM, ATTN: BENÉT LABORATORIES, SMCAR-CCB-TL, WATERVLIET, NY 12189-4050 OF ADDRESS CHANGES.